Info
Februarie 1998
Vol. 8/1

Lei 5000

Gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

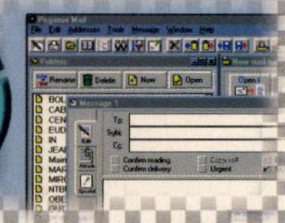
Focus:
Prima lecție de JAVA
PAG. 18

Frontiere:
Un meci de șah istoric
PAG. 30

Internet:
Scrisori în eter
PAG. 48

Babel:
Adevărul despre LOGO
PAG. 34

Profil:
Povestea lui Linus Torvalds
PAG. 46



Plus:

Probleme și rezolvări „olimpice”
PAG. 4
Matematica și tipurile de date
PAG. 24

T.P.
Aprobat R.A.P.R.
Nr.103/DC/053/1998
Valabil - 1998 -

INFADRES

aveți
• un domeniu de activitate
• mai multe birouri/sedii
• angajați
• multe hârtii și probleme de rezolvat
HARABABURĂ !!!
vă doriți
• echipamente hardware interconectate
• software de interfață & de aplicație
• școlarizarea angajaților
ORDINE !!!
vă oferim
• soluții hardware & software
• sisteme integrate "la cheie"
• asistență tehnică software & service hardware
• consultanță
• training
cartea noastră de vizită
• System Integration & Project Management - proiecte la nivel național
• IBM Business Partner
• Microsoft Authorized Distribution Partner/ DSP for Romania & Microsoft Hot Line
• distribuitor Symantec
• distribuitor de PC-uri FORTE
• 9 sucursale FORTE
Microsoft
WHERE DO YOU WANT TO GO TODAY ?™
Authorized Distribution Partner/
DSP for Romania
FORTE
COMPUTERS
Flexibility.Made in Romania
IBM
Lipscani 102, București
Tel: 3122360/6137282 Fax: 3122630 Microsoft Hot Line: 3120948

Vol. 8, Nr. 1, Februarie 1998

gazeta de informatică

Redacția:

Director științific:
conf. dr. *Horia Georgescu* (București)
Redactor șef:
prof. *Clara Ionescu* (Cluj)
Editor executiv:
Mircea Sârbu (Tg-Mureș)

Au mai colaborat:
Ionuț Mușlea (Cluj)
Mihai Budiu (București)
prof. *Emil Onea* (Focșani)
Gabriel Proșcanu (București)
Iosif Fettich (Tg-Mureș)

Proiectul machetei:
Octavian Lipovan
Mircea Sârbu
Coperta și grafica:
Adi Pop

Adresa pentru corespondență:
C.P. 172-1, 4300 Tg-Mureș
Adresa e-mail: office@ginfo.ro
Tel: (065) 166.516
Fax: (065) 166.290

ISSN: 1222-7129
Tiparul: *S.C. Infopress srl*,
Odorheiu Secuiesc,
tel: 066.216.483

Editor: Computer Press Agora srl
str. T. Vladimirescu nr. 63/1
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director editorial:
Mircea Sârbu
Director tehnic:
Szabó László
Reprezentanța București:
Director: *Dana Pastramă*
Tel: (01) 330.92.81

Sumar

editorial.. PAG. 3

Hârtie și cerneală
de Mircea Sârbu
Câte ceva despre noua formulă a Gazetei de Informatică și despre planurile de viitor.

soluții.. PAG. 4

probleme REZOLVATE
pagini realizate de Clara Ionescu
Patru probleme de la Olimpiada Națională, Timișoara, 1997.

probleme.. PAG. 11

Propuse pentru REZOLVARE
***
La Poiana Pinului, olimpicii au rezolvat și au compus probleme.

focus.. PAG. 18

Structuri înlănțuite în limbajul JAVA
de Horia Georgescu
Un limbaj în care lumea informaticii își pune mari speranțe. Prima lecție.

PAG. 24

TIPURI de DATE
de Mihai Budiu
Credeați că știți totul despre tipurile de date?

frontiere.. PAG. 30

KASPARO V învins de DEEP BLUE...
de Ionuț Mușlea
Prima victorie a unui computer asupra unui campion mondial.

babel.. PAG. 34

LOGO — un LIMBAJ serios?
de Mircea Sârbu
A fost conceput pentru copii. Dar poate fi folosit de oricine.

profil.. PAG. 46

The NEVER ENDING Story
de Gabriel Proșcanu
Copilul-minune al informaticii: Linus Torvalds, creatorul lui Linux.

internet.. PAG. 48

E-MAIL: mesaje în eter
de Mircea Sârbu
Principiile de bază coincid cu cele ale poștei clasice. Nu și detaliile.

dialog.. PAG. 60

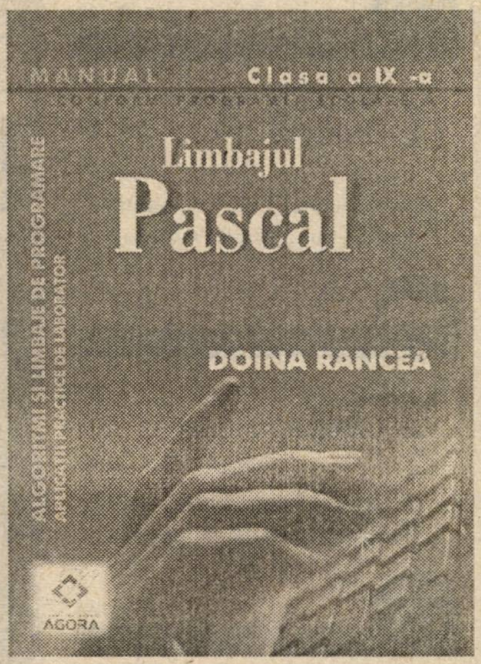
POȘTA redacției

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie*

... un MANUAL.



Limbajul Pascal este un manual care acoperă programa școlară (1997) a două discipline: „Algoritmi și limbaje de programare" și „Aplicații practice de laborator" care se studiază în clasele a IX-a cu profil de informatică.

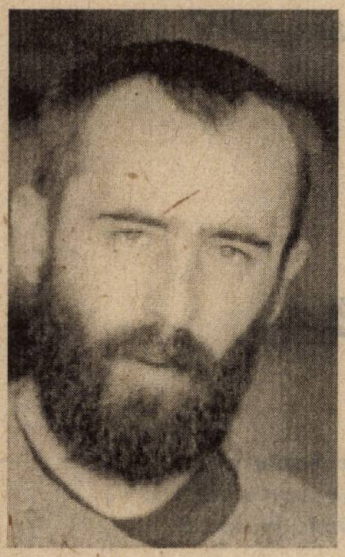
Manualul prezintă noțiunile din programă la un nivel accesibil, fiecare noțiune fiind ilustrată în mai multe programe Borland Pascal 7.0. Aceste programe sunt însoțite și de prezentarea datelor de intrare și a rezultatelor în forma sub care apar pe ecran. Problemele au fost selectate în așa fel încât să ilustreze multiplele domenii de utilizare a calculatoarelor. Nu lipsesc problemele de simulare, jocuri, probleme distractive, dar au fost alese și aplicații din programa de matematică de clasa a IX-a cu scopul exemplificării și fixării unor noțiuni importante din matematică. În manual se insistă asupra felului în care se descoperă rezolvarea unei probleme, nivelul de dificultate a acestora crescând gradat, venind astfel în sprijinul elevilor și profesorilor.
Numeroasele probleme din manual permit utilizarea acestuia nu numai pentru studiu individual ci și ca bogat material didactic pentru lecțiile de programare și laborator.
Manualul se adresează elevilor, studenților, profesorilor și oricărei persoane interesate de programare.
Manualul a fost conceput, plecând de la ediția din 1993 (Limbajul Turbo Pascal, clasa a X-a) care a fost regândit substanțial și îmbogățit, astfel încât să fie accesibil unui elev de clasa a IX-a. Ediția anterioară s-a bucurat de un succes deosebit, fiind apreciată de mii de cititori.

Autoarea, Doina Rancea, este absolventă cu diplomă de merit a Facultății de Matematică, Secția Informatică a Universității Babeș-Bolyai din Cluj Napoca (1975). A fost repartizată la Liceul de Informatică din Cluj Napoca unde predă și în prezent limbajele de programare Pascal și C++. Este profesoară gradul I și în această calitate îndrumă activitatea de practică pedagogică a studenților. Experiența bogată a autoarei a fost valorificată cu ocazia participării, în cadrul Comisiei Naționale de Specialitate Informatică a Ministerului Educației Naționale, la actualizări ale programei școlare pentru disciplina informatică și în cadrul comisiilor județene și naționale ale olimpiadelor, propunând probleme originale.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

HÂRTIE și CERNEALĂ



La vremea când scriu aceste rânduri, revista pe care o ai acum în mână nu există. Nu există ca obiect. Nu pot încă să-i simt mirosul amestecat de hârtie și cerneală, aroma de tipăritură proaspătă. În schimb, există undeva în calculatoarele cu care a fost făcută și, mai ales, există în mintea celor care au conceput-o. Făcând parte dintre aceștia, am să încerc să te pun și pe tine în temă. Pentru că o revistă trăiește de fapt prin tine, cel care o citești.

Continuitate

Nu, nu este o revistă nouă. Dacă privești mai atent coperta, vei observa că scrie „volumul 8, nr. 1". Dacă nu știai, află acum că Gazeta de Informatică apare încă din 1991, datorită muncii, perseverenței și, mai cu seamă, pasiunii unui grup de oameni dintre care voi menționa acum pe cei fără a căror efort revista ar fi murit de mult: domnul conf. dr. Horia Georgescu, profesor la Facultatea de Matematică a Universității București, și doamna Clara Ionescu de la Liceul de Informatică din Cluj.

Numele lor apare și acum în caseta redacției și reprezintă, cred eu, o garanție că revista va duce mai departe un standard calitativ și o anumită tematică. Așadar, vei găsi în continuare pagini cuprinzând soluțiile unor probleme date la diverse concursuri (secțiunea *Soluții*), pagini cu probleme propuse spre rezolvare (secțiunea *Probleme*), precum și articole despre algoritmică și programare (secțiunea *Focus*).

Ce e nou?

Nou este, după cum ai observat deja, formatul și designul. Poate n-ai observat în schimb că nou este și editorul. Pe scurt, istoria este următoarea: între editura Libris (vechiul editor) și Computer Press Agora (actualul editor) a survenit un contract de colaborare. Cum Computer Press Agora editează deja alte trei reviste de specialitate și dispune de întreaga infrastructură necesară, a preluat și GInfo, în ideea de a-i întări poziția și a-i extinde audiența.

Ideea comună a fost de a crea revistei un nou *look*, mai modern și mai atractiv (ce spui, este?) și de a-i extinde aria tematică și spre domenii de mare interes, care însă nu sunt cuprinse încă în programa școlară pentru clasele de liceu. Astfel, au apărut câteva secțiuni noi.

Secțiunea *Frontiere* își propune să găzduiască în fiecare număr câteva pagini dedicate unor realizări de mare însemnătate din domeniul informaticii. Cum ar fi, de pildă, performanța – până mai ieri de neimaginat – realizată de un computer-șahist conceput de IBM: victoria în fața campionului mondial.

Universului limbajelor de programare îi este dedicată secțiunea *Babel*. Aici ne propunem să-ți prezentăm foarte rapid niște limbaje mai „ciudate", mai „exotice" sau, pur și simplu, mai altfel.

Istoria informaticii este scurtă, dar tumultoasă. Iar cum istoria o fac personalitățile, ne-am gândit că poate te interesează să afli mai multe despre marii învingători sau marii învinși ai unei științe și a unei industrii care se dezvoltă într-un ritm amețitor.

În fine, domeniul cu cea mai rapidă dezvoltare: Internetul. Cu speranța că în scurt timp toți cititorii noștri vor avea acces la această uriașă resursă de informare, de comunicare și – foarte curând – de calcul, credem că Rețeaua nu trebuie să te găsească nepregătit...

Ce mai urmează?

Mai avem unele planuri pe care le vom transpune în practică în lunile următoare. Acestea vizează câteva noi secțiuni, printre care una care să prezinte actualități de interes pentru pasionații de informatică și o altă secțiune care să răscolească Web-ul în căutarea de pagini interesante, nostime, instructive...

O altă intenție este de a publica în fiecare număr câte o lucrare realizată de un elev și prezentată la un simpozion, colocviu sau concurs. Sugestiile tale no vor fi de mare ajutor.

Concursurile deja încep (vezi și *Poșta Redacției*), dar amploarea pe care cu siguranță ți-o dorești o vor dobândi abia din toamnă, odată cu începerea anului școlar. Până atunci vom pune la punct canalele de distribuție și vom căuta sponsorii care să ofere marile premii. Dar mai ales, vom încerca să aducem revista tot mai aproape de tine.

Programul apariților

Ritmul revistei va fi legat de ritmul școlii. GInfo va avea opt numere anual: septembrie - decembrie (apare la sfârșitul lunii), februarie - mai (când va apărea la începutul lunii). Abonamentele se vor face pe ani școlari (deci sept. - iun.).

Revista va avea și o pagină Web, începând chiar din luna februarie. Pe parcurs, vom face din această pagină o extensie a revistei. Ea va cuprinde materiale adiționale legate de articolele apărute (surse, specificații etc.), articole inedite și, cu timpul, se vor forma și arhivele.

Însă oricât de *cool* ar fi paginile Web, ele nu vor avea niciodată mirosul cel minunat de hârtie și cerneală, de tipăritură proaspătă. Nu-i așa? ❑

Mircea Sârbu

SOLUȚIILE problemelor propuse

În Gazeta de Informatică. nr.2/1997 au fost publicate subiectele Olimpiadei de Informatică. Reamintim - sub o formă mai scurtă - enunțurile problemelor ale căror rezolvări sunt prezentate în acest număr.

Clasa a IX-a: Baze ascunse

Problemă propusă de prof. Ioan Maxim, Liceul de Informatică, Suceava.

Enunț:

În fișierul `INPUT.TXT` se dau `n` numere naturale $x_1, x_2, \ldots, x_n$ având fiecare cel mult 9 cifre. Numărul x_i este reprezentat într-o bază b_i ($2 \le b_i \le 10$), care nu este cunoscută. Să se determine pentru fiecare număr x_i o bază b_i, astfel încât intervalul de valori `[a,b]` în care sunt cuprinse toate cele `n` numere să fie de lungime minimă. Prin lungimea unui interval se înțelege diferența în valoare absolută dintre capetele intervalului.

Exemplu:

INPUT.TXT:	OUTPUT.TXT
5	35 6
635 27 100 13 19	27 8
	100 4
	13 10
	19 10
	13 23

Rezolvare:

Rezolvarea propusă de **Mihai Oltean**, student, folosește următorul algoritm (bazat pe o idee aparținând unui prieten al autorului problemei):

1) Pentru fiecare număr `x[i]` se determină bazele în care este posibil ca acesta să fie reprezentat.

2) Se trece fiecare număr din bazele presupuse posibile în baza `10`, rezultatul păstrându-se într-un șir `b`. Acest șir va conține reprezentările în baza `10` ale numerelor din șirul inițial, considerate ca fiind reprezentate în oricare din bazele posibile.

3) Se ordonează șirul obținut.

4) Se determină secvența de lungime minimă care conține fiecare număr din șirul inițial cel puțin o dată. Prin lungimea secvenței se înțelege diferența dintre numerele care formează capetele ei.

Listing: BAZE.PAS

```
Program Baze_ascunse;
const MaxN=1000;
type vect=array[1..MaxN] of Byte;
var i,j,n,sr,st:Byte;
    a:array[1..MaxN] of Longint;
    b:array[1..9*MaxN] of Longint;
    c:array[1..9*MaxN] of Byte;
    nrpos,tmp:vect;
    max:Char; total:Word;
    s:string;

  procedure Citeste;
  var f:text;
  begin
    Assign(f,'input.txt'); Reset(f); Readln(f,n);
    for i:=1 to n do
      if not Seekeoln(f) then
        Read(f,a[i]);
    Close(f)
  end;

  Function Trans(numar:Longint; baza:Byte):Longint;
  var nr,p:Longint;
  begin
    p:=1; nr:=0;
    while numar>0 do
    begin
      nr:=nr+p*(numar mod 10);
      numar:=numar div 10;
      p:=p*baza
    end;
    trans:=nr
  end;
```

```
procedure Sort(L,R:Integer);
var i,j:Integer; x,y:Longint;
begin
  i:=L; j:=R; x:=b[(L+R) div 2];
  repeat
    while b[i]<x do i:=i+1;
    while x<b[j] do j:=j-1;
    if i<=j then
      begin
        y:=b[i]; b[i]:=b[j]; b[j]:=y;
        y:=c[i]; c[i]:=c[j]; c[j]:=y;
        i:=i+1; j:=j-1
      end
  until i>j;
  if L<j then Sort(L,j);
  if i<R then Sort(i,R)
end;

function posibil:Boolean;
var g:Byte;
begin
  for g:=1 to n do
    if nrpos[g]=0 then
      begin
        posibil:=false;
        Exit
      end;
  posibil:=true
end;

procedure DeterminaSecventa;
var gasit:Boolean;
    start,stop:Byte;
    interval:Longint;
begin
  gasit:=true; start:=1; interval:=1000000000;
  while gasit and (st<=total) and posibil do
  begin
    stop:=total; gasit:=false; tmp:=nrpos;
    while (stop>=start) and posibil do
      begin
        stop:=stop-1;
        gasit:=true;
        Dec(nrpos[c[stop+1]])
      end;
    if gasit
    then if interval>stop-start+2 then
        begin
          interval:=b[stop+1]-b[start-1];
          st:=stop+1; sr:=start-1
        end;
    Dec(tmp[c[start]]);
    nrpos:=tmp; start:=start+1
  end
end;
```

```
procedure ReconstituieSecventa;
var i,j,bb:Word;
    f:Text; gasit:Boolean;
begin
  Assign(f,'output.txt'); Rewrite(f);
  for j:=1 to n do
  begin
    gasit:=false;
    for i:=sr to st do
      if (c[i]=j) and not gasit
      then
      begin
        bb:=10; gasit:=true;
        while trans(a[c[i]],bb)<>b[i] do Dec(bb);
        Writeln(f,a[j],' ',bb)
      end
  end;
  Close(f)
end;

Begin
  citeste; total:=0;
  for i:=1 to n do
  begin
    Str(a[i],s); max:='1';
    for j:=1 to Length(s) do
      if max<s[j] then max:=s[j];
    nrpos[i]:=10-(Ord(max)-Ord('0'));
    for j:=Ord(max)-Ord('0')+1 to 10 do
    begin
      Inc(total);
      b[total]:=trans(a[i],j);
      c[total]:=i
    end
  end;
  Sort(1,total);
  DeterminaSecventa;
  ReconstituieSecventa
End.
```

Clasa a X-a: **Alianțe politice**

Problemă propusă de lector dr. Marin Popa, Universitatea București.

Enunț:

Spunem că un partid x simpatizează un alt partid y dacă în programul politic al partidului y se regăsesc idei ale programului partidului x. Numim coaliție de partide o grupare de două sau mai multe partide în care orice două partide se simpatizează reciproc.

Fiind date n partide (n≤20, dat în fișierul FIS.INT) să se determine coaliția cu cel mai mare număr de partide politice. Dacă sunt mai multe cu acest număr maxim, atunci să se afișeze toate.

Dacă partidul i simpatizează cu j, în fișierul de intrare în linia i, pe poziția j este scris 1, altfel în fișier este scris 0.

Exemplu:

FIS.INT:	FIS.OUT:
6	1 2 4 6
0 1 0 1 0 1	
1 0 1 1 0 1	
0 1 0 1 0 0	
1 1 0 0 1 1	
0 1 0 1 0 0	
1 1 0 1 1 0	

Rezolvare:

Mihai Oltean, student.

Cele n partide politice pot fi considerate ca fiind nodurile unui graf în care nodul i este adiacent cu nodul j dacă partidul i simpatizează partidul j.

Interpretând în termenii teoriei grafurilor cerințele problemei, rezultă că trebuie determinate toate subgrafurile maximale complete formate din cel puțin două vârfuri.

Propunem o rezolvare prin metoda backtracking: primul parametru (k) al procedurii back contorizează numărul de noduri adăugate până la momentul curent în tabloul sol care conține o soluție parțială a problemei; al doilea parametru este folosit pentru a optimiza implementarea metodei, și anume el va reține vecinii nodurilor care formează soluția parțială construită până la pasul curent.

Soluția este construită în tabloul sol, soluțiile finale fiind reținute în tabloul solfin. Totodată pentru o mai ușoară prelucrare, soluția se mai reține și în variabila multime de tip **set**.

Variabila ordin reprezintă numărul de noduri ale subgrafului maximal complet. Inițial această variabilă are valoarea n, dar programul se poate optimiza, calculând în funcție de numărul de muchii o limită superioară de la care această variabilă trebuie să descrească.

Listing: POLITIC.PAS

```
Program Aliante_politice;
type solutie=array[1..20] of Byte;
     mult=set of 1..20;
var a:array[1..20,1..20] of Byte;
    vec:array[1..20] of mult;
    sol:solutie;
    solfin:array[1..10] of solutie;
    n,i,j,ordin,nr:Byte;
    gasit:Boolean; multime:mult; f:Text;

  procedure citeste;
  var c:Char;
  begin
    Assign(f,'fis.int'); Reset(f); Readln(f,n);
    for i:=1 to n do
    begin vec[i]:=[];
      for j:=1 to n do
      begin Read(f,c);
            a[i,j]:=Ord(c)-48;
            if a[i,j]=1 then vec[i]:=vec[i]+[j] end;
          Readln(f)
    end;
    Close(f)
  end;

  procedure back(k:Byte; mv:mult);
  var v:Byte;
  begin
    if k=ordin+1 then
      begin
        Inc(nr); solfin[nr]:=sol; gasit:=true
      end
    else
    begin
      if k=1
      then
        for v:=1 to n-ordin+1 do
        begin
          multime:=multime+[v]; sol[k]:=v;
          back(k+1,vec[v]); multime:=multime-[v]
        end
      else
        for v:=sol[k-1]+1 to n-ordin+k do
        if (vec[v]*multime=multime) and (v in mv)
        then
        begin
          multime:=multime+[v]; sol[k]:=v;
          back(k+1,mv*vec[v]); multime:=multime-[v]
        end
    end
  end;

  procedure afis;
  begin
    Assign(f,'fis.out'); Rewrite(f);
    for i:=1 to nr do
    begin
      for j:=1 to ordin do Write(f,solfin[i,j],' ');
      Writeln(f)
    end;
    Close(f)
  end;

Begin
  citeste;
  for ordin:=n downto 2 do
  begin
    nr:=0; gasit:=false; back(1,[]);
    if gasit then begin afis; Halt end
  end;
  Assign(f,'output.txt'); Rewrite(f);
  Writeln(f,'Nu exista coalitie.');
  Close(f)
End.
```

Clasa a XI-a: Trepte

Problemă propusă de prof. Clara Ionescu, Liceul de Informatică, Cluj.

Enunț:

În Estul Îndepărtat, pe vârful unui munte, se va construi un templu la care vor urca trepte având dimensiuni diferite. Proiectantul trebuie să determine toate variantele în care aceste trepte se pot construi, astfel încât numărul lor să fie cel mai mare posibil.

Înălțimea totală ($2 < I \leq 50$) respectiv lățimea totală ($2 < L \leq 50$) pe care se vor întinde treptele, se citesc din fișierul de intrare `INPUT.TXT`. Fiecare treaptă având o înălțime și o lățime, se cere ca oricare două înălțimi și oricare două lățimi să fie diferite, de asemenea oricare înălțime trebuie să difere de oricare lățime. Nici o înălțime nu poate fi mai mare decât o anumită valoare dată M ($1 \leq M \leq 12$).

Programul trebuie să determine numărul maxim de trepte posibile, de asemenea toate variantele care nu sunt „identice" în care poate fi realizat proiectul. Două variante se consideră „identice" dacă în acestea se regăsesc aceleași înălțimi (eventual în altă ordine) și aceleași lățimi (eventual în altă ordine). Dacă nu se poate determina nici un proiect care să respecte cerințele puse, în fișierul de ieșire se va scrie mesajul: `'NU'`

Exemplu:

INPUT.TXT	OUTPUT.TXT
8	2
5	1 7
7	2 3
	2 6
	1 4
	3 5
	1 4

Rezolvare:

Mihai Scorțaru, clasa XI-a, Liceul de Informatică, Cluj

Problema se reduce la determinarea tuturor partițiilor cu numere distincte, atât pentru lungimea treptelor cât și pentru înălțimea lor, respectând condițiile ca nici o treaptă să nu aibă înălțimea mai mare decât înălțimea maximă admisă și nici o lungime să nu fie egală cu vreo înălțime.

Pentru determinarea partițiilor vom folosi metoda backtracking, implementată recursiv; din cauza condiției suplimentare în cazul înălțimilor, vom avea două proceduri diferite.

Partițiile se vor păstra în articole care vor conține numărul de elemente al partiției precum și partiția propriu-zisă. Partiția propriu-zisă poate fi reprezentată într-un vector cu doar 7 elemente deoarece dacă alegem 8 numere a căror sumă să fie 50 (maximul pentru înălțime și lungime) nu vom putea alege alte 8 numere a căror sumă să fie cel mult 50.

După ce păstrăm partițiile celor două numere, le testăm două câte două. În studiul unei perechi de partiții vom avea următoarele cazuri:

- Perechea nu poate fi soluție ⇒ ignorăm perechea;
- Perechea poate fi soluție dar cardinalul ei e inferior cardinalului celei mai bune soluții găsite până acum ⇒ ignorăm perechea;
- Perechea poate fi soluție și cardinalul ei este egal cu cardinalul celei mai bune soluții găsite până acum ⇒ scriem perechea în fișierul de ieșire;
- Perechea poate fi soluție și cardinalul ei este mai mare decât cardinalul celei mai bune soluții determinate până acum ⇒ ștergem fișierul de ieșire și scriem cardinalul perechii urmat de perechea care devine cea mai bună soluție de până acum.

În cazul în care cardinalul celei mai bune soluții este mai mic decât 2 ștergem fișierul de ieșire și scriem în el `'NU'`.

Listing: TREPTE.PAS

```
Program Trepte;
type partitie=record
                nrelem:Byte;
                elem:array[1..7]of Byte
              end;
var f:Text;
    x:array[0..9]of Byte;
    tot1,tot2,total:Word;
    a,b:array[1..3658] of partitie;
    H,L,M:Byte;

  procedure part(ind:Byte; suma:Byte);
  var i,j:Byte;
  begin
    if ind > 7 then Exit;
    for i:=x[ind-1]+1 to L-suma do
    begin
      x[ind]:=i;
      if suma+i=L then
        begin
          Inc(total);
          with a[total] do
            begin
              nrelem:=ind;
              for j:=1 to ind do
                elem[j]:=x[j]
            end
        end
      else part(ind+1,suma+i)
    end
  end;
```

```
procedure part2(ind:Byte; suma:Byte);
var i,j:Byte;
begin
  if ind > 7 then Exit;
  for i:=x[ind-1]+1 to H-suma do
  begin
    if i > M then Exit;
    x[ind]:=i;
    if suma+i=H
    then begin Inc(total);
               with b[total] do
               begin nrelem:=ind;
                 for j:=1 to ind do elem[j]:=x[j]
               end
         end
    else part2(ind+1,suma+i)
  end
end;

procedure genpart;
begin
  total:=0; part(1,0);
  tot1:=total; total:=0; part2(1,0);
  tot2:=total
end;

procedure citire;
begin
  Assign(f,'input.txt'); Reset(f);
  Readln(f,H,L,M); Close(f)
end;

function bun(i,j:Word):Boolean;
var k,L:Byte;
begin
  bun:=false;
  for k:=1 to a[i].nrelem do
    for L:=1 to a[i].nrelem do
      if a[i].elem[k]=b[j].elem[l] then Exit;
  bun:=true
end;

procedure scriere;
var i,j,max,k:Word;
begin
  Assign(f,'output.txt'); Rewrite(f); max:=1;
  for i:=1 to tot1 do
    if a[i].nrelem>=max then
      for j:=1 to tot2 do
        if b[j].nrelem=a[i].nrelem then
          if bun(i,j) then
          begin
            if max<a[i].nrelem then
            begin Close(f); Rewrite(f);
                  max:=a[i].nrelem; Writeln(f,max)
            end;
            for k:=1 to max do
              Write(f,b[j].elem[k],' ');
              Writeln(f);
            for k:=1 to max do
              Write(f,a[i].elem[k],' ');
              Writeln(f); Writeln(f)
          end;
  Close(f);
  if max < 2 then
    begin
      Rewrite(f); Writeln(f,'NU'); Close(f)
    end
end;

Begin
  citire;
  genpart;
  scriere
End.
```

Clasa a XII-a: **Beculețe**

Problemă propusă de prof. Emanuela Mateescu, Liceul de Informatică, Iași

Enunț:

Fie o rețea ortogonală ale cărei linii sunt la distanța unitară, iar nodurile ei sunt în puncte de coordonate întregi aparținând intervalului [-16000,16000]; pătratele rețelei sunt colorate inițial în alb (culoarea de indice 0).

În n noduri ale rețelei ($1 \le n \le 255$) se află câte un beculeț. Unind în linie dreaptă, printr-un fir, două beculețe, toate pătratele traversate efectiv de fir (firul trece prin interiorul pătratului) își schimbă culoarea din culoarea i în culoarea i+1. Instalația funcționează dacă și numai dacă între oricare două becuri există (direct sau indirect) legătură.

Determinați o modalitate de conectare a celor n becuri, astfel încât instalația să funcționeze, iar coloratura rețelei să fie de indice minim. Prin indicele unei coloraturi înțelegem suma indicilor de culoare a tuturor pătratelor din rețea.

În fișierul de ieșire OUTPUT.TXT se vor scrie: indicele coloraturii obținute, numărul legăturilor directe realizate și coordonatele a două beculețe unite prin fir direct, separate prin câte un spațiu.

Exemple:

1) INPUT.TXT:	OUTPUT.TXT:
3	56
0 0	2
20 5	0 0 20 5
18 40	20 5 18 40

2) INPUT.TXT:	OUTPUT.TXT:
5	12
0 0	4
2 10	2 10 5 7
1 5	1 5 0 0
5 7	5 7 1 5
10 0	10 0 0 0

Rezolvare:

Mihai Oltean, student.

Se consideră cele n puncte ca fiind nodurile unui graf (planar). Se definește ponderea (lungimea) unei muchii ca fiind numărul de pătrate ale căror interioare le străbate. Pentru a calcula acest număr ne vom folosi de următoarele proprietăți (*Gazeta de Matematică*, nr. 11/1987):

Definiție:

Numim puncte laticiale, punctele de coordonate întregi în plan.

Proprietatea 1:

Fie $a, b \in \mathbb{N}$ și $f:[a,b] \to \mathbb{R}$ o funcție continuă și pozitivă. Considerăm domeniul D mărginit de axa Ox, dreptele $x=a$, $x=b$ și graficul lui f. Atunci numărul punctelor laticiale din D este dat de expresia:

$$b-a+1+\sum_{k=a}^{b}[f(k)]$$

unde cu [ ] s-a notat funcția partea întreagă.

Demonstrație:

Toate punctele laticiale din D se află pe dreptele $x=k$ cu $a \le k \le b$, $k \in \mathbb{N}$. Numărul punctelor laticiale de pe segmentul de dreaptă determinat de intersecția dreptei $x=k$ cu domeniul D este $1+[f(k)]$. Numărul căutat se obține însumând valorile acestei expresii pentru $k=a$, $a+1$, $a+2,\ldots,b$. Obținem:

$$\sum_{k=a}^{b}(1+[f(k)])=b-a+1+\sum_{k=a}^{b}[f(k)].$$

Proprietatea 2:

Numărul de puncte laticiale de pe segmentul de dreaptă determinat de punctele $O(0,0)$ și $M(m,n)$ este de $(m,n)+1$ unde prin (m,n) s-a notat c.m.m.d.c. al lui m și n.

Demonstrație:

Fie $d=(m,n)$. Atunci $(\exists)m_1$ și $n_1 \in \mathbb{N}$ astfel încât $m=dm_1$, $n=dn_1$ și $(m_1,n_1)=1$.

Segmentul OM este determinat de dreapta de ecuație $y=\frac{n_1}{m_1}x$ cu $x \in [0,m]$. Se verifică ușor că punctele P_k (km_1,kn_1) pentru $k=0,1,2,\ldots,d$ sunt puncte laticiale pe OM. Să demonstrăm că ele sunt singurele. Presupunem că $Q(m_0,n_0)$ este punct laticial de pe segmentul OM astfel încât $Q \notin \{Pk \mid k=0,1,2,\ldots,d\}$, (*).

Deci $m_0,n_0 \in \mathbb{N}$. Ele satisfac ecuația $y=\frac{n_1}{m_1}x$ și $m_0 \in [0,m]$. Avem $n_0=\frac{n_1}{m_1}\cdot m_0$.

Ținând cont de faptul că $n_0 \in \mathbb{N}$ și că $(n_1,m_1)=1$, rezultă că $m_1 | m_0$. Deci $m_0=km_0$ unde $k \in \mathbb{N}$ și $0 \le k \le h$ (deoarece $m_0 \in [0,m]$).

Avem $n_0=\frac{n_1}{m_1}\cdot m_0=\frac{n_1}{m_1}\cdot km_1=kn_1$.

Deci coordonatele lui Q se scriu:

$$\begin{cases} m_0=km_1 \\ n_0=kn_1 \end{cases} \text{ unde } 0 \le k \le d.$$

Rezultă că $Q \in \{P_k \mid k=0,1,2,\ldots,d\}$ ceea ce contrazice relația (*).

Deci singurele puncte laticiale de pe segmentul OM sunt P_k cu $k=0,1,2,\ldots,d$. Evident, ele sunt în număr de $d+1=(m,n)$.

Având la bază aceste noțiuni, afirmăm că numărul de pătrate ale căror interioare le străbate dreapta care unește punctele `(0,0)` și `(n,m)` este egal cu `n+m-cmmdc(n,m)`.

Demonstrarea acestei afirmații rezultă din:

1) O dreaptă care nu trece prin nici un punct laticial (exceptând extremitățile) la ieșirea din interiorul unui pătrat va intra în interiorul unuia din pătratele de deasupra, dedesubt, stânga sau dreapta.

2) Pentru fiecare punct laticial nou, pătratele învecinate cu el și ale căror interioare sunt străbătute de dreapta respectivă sunt învecinate pe diagonală.

Din enunțul problemei se deduce că trebuie să se determine un arbore parțial de cost minim. Datorită faptului că numărul de pătrate ale căror interioare sunt străbătute de o dreaptă paralelă cu axele de coordonate, în momentul în care se construiește arborele, este egal cu `0`, este necesar ca în cazul în care trebuie ales între mai multe muchii cu aceeași pondere să se aleagă aceea care are lungimea (euclidiană) cea mai mică.

În programul care implementează acest algoritm, informațiile despre muchii sunt reținute în Heap. Deci se va construi o listă (procedura `ConstruiesteLista`) care are `n*(n-1)/2` noduri. Informația atașată unui nod al listei constă din numerele de ordine (ordinea din fișierul de intrare) ale capetelor muchiei și numărul de pătrate pe care le străbate muchia.

Pentru determinarea arborelui parțial de cost minim s-a folosit algoritmul lui Prim. Acest arbore este memorat în tabloul `b`, iar datele de intrare sunt citite în vectorul `a`.

Listing: BEC.PAS

```
Program Beculete;
type nod=record x,y:Longint end;
     lista=^art;
     art=record
          n1,n2:Byte; lung:Word;
          urm:lista
        end;
var n,i,j:Integer; cost:Longint;
    cap:lista;
    a:array[1..255] of nod;
    b:array[1..254,1..2] of Byte;
```

```
procedure citeste;
var f:Text;
begin
  Assign(f,'input.txt'); Reset(f); Readln(f,n);
  for i:=1 to n do Readln(f,a[i].x,a[i].y);
  Close(f)
end;

function cmmdc(q,w:Integer):Integer;
begin
  if w=0 then cmmdc:=q
         else cmmdc:=cmmdc(w,q mod w)
end;

procedure ConstruiesteLista;
var p,r:lista;
begin
  cap:=nil; new(cap);
  cap^.n1:=1; cap^.n2:=2;
  cap^.lung:=Abs(a[1].x-a[2].x)+Abs(a[1].y-a[2].y)-
       cmmdc(Abs(a[1].x-a[2].x),Abs(a[1].y-a[2].y));
  p:=cap;
  for i:=1 to n-1 do
    for j:=i+1 to n do
    begin
      new(r); r^.n1:=i; r^.n2:=j;
      r^.lung:=Abs(a[i].x-a[j].x)+
               Abs(a[i].y-a[j].y)-
               cmmdc(Abs(a[i].x-a[j].x),
                     Abs(a[i].y-a[j].y));
      r^.urm:=nil; p^.urm:=r; p:=r
    end;
  cap:=cap^.urm
end;

function CalcDistEucl(p:lista):Longint;
begin
  CalcDistEucl:=
    (a[p^.n1].x-a[p^.n2].x)*(a[p^.n1].x-a[p^.n2].x)+
    (a[p^.n1].y-a[p^.n2].y)*(a[p^.n1].y-a[p^.n2].y)
end;

procedure ConstruiesteArbore;
var vizitate:set of Byte;
    nr,v1,v2,min:Word; minre:Longint;
    p:lista;
begin
  vizitate:=[1];
  nr:=0;
  cost:=0;
  while vizitate<>[1..n] do
  begin
    p:=cap;
    min:=64000;
    minre:=MaxLongint;
    while p<>nil do
    begin
      if ((p^.n1 in vizitate) and
          not (p^.n2 in vizitate)) or
         ((p^.n2 in vizitate) and
          not (p^.n1 in vizitate))
      then if min>p^.lung
        then
          begin
            min:=p^.lung;
            minre:= CalcDistEucl(p);
            v1:=p^.n1; v2:=p^.n2
          end
        else
          if (min=p^.lung)
              and (minre>CalcDistEucl(p))
          then
          begin
            minre:= CalcDistEucl(p);
            v1:=p^.n1; v2:=p^.n2
          end;
      p:=p^.urm
    end;
    Inc(nr);
    cost:=cost+min;
    b[nr,1]:=v1; b[nr,2]:=v2;
    vizitate:=vizitate+[v1,v2]
  end
end;

procedure ListeazaArbore;
var f:Text;
begin
  Assign(f,'output.txt'); Rewrite(f);
  Writeln(f,cost); Writeln(f,n-1);
  for i:=1 to n-1 do
  begin Write(f,a[b[i,1]].x,' ',a[b[i,1]].y,' ');
        Writeln(a[b[i,2]].x,' ',a[b[i,2]].y)
  end;
  Close(f)
end;

Begin
  citeste;
  ConstruiesteLista;
  ConstruiesteArbore;
  ListeazaArbore
End.
```

Programe sursă:

Puteți descărca listingurile prezentate prin Internet de la adresa http://www.ginfo.ro/8_1/surse

Paginile „Soluții" au fost realizate de **Clara Ionescu**.
E-mail: cionescu@ginfo.ro.

PROBLEME propuse pentru REZOLVARE

La începutul lunii ianuarie s-a desfășurat la Poiana Pinului (Buzău) o tabără de pregătire și odihnă a elevilor olimpici. Aceștia s-au odihnit rezolvând și propunând probleme. Prin amabilitatea d-lui prof. Emil Onea (Focșani) avem posibilitatea să vă oferim câteva dintre ele. Așteptăm rezolvări. Succes!

P019801: Constelații

Propusă de Cristian Cadar, elev la Colegiul Național „Sf. Sava", București.

Enunț:

Într-o seară petrecută la „Poiana Pinului", un informatician admiră cerul și compune următoarea problemă:

Se dau m constelații cu minimum 3 și maximum n stele fiecare. Se definește „bordura" unei constelații ca fiind acoperitoarea convexă a stelelor din respectiva constelație. Fiecare stea este dată prin coordonatele sale (x, y) din „planul stelar". De asemenea, se dă poziția Lunii tot prin cele două coordonate ale sale.

Se cere:

a) să se afle care constelații conțin Luna în interiorul bordurii sau pe bordura acestora;

b) să se determine (dacă există) o constelație în a cărei bordură se încadrează toate celelalte borduri de constelații.

Limite: $1 \le m \le 100$ și $1 \le n \le 1000$

Datele de intrare se citesc din fișierul `STELE.IN` în următorul format:

`X Y`	→ coordonatele Lunii
`m`	→ nr. de constelații
N_1	→ nr. de stele din prima constelație
x_{11} y_{11}	→ coordonatele primei stele din prima constelație
.....	
x_{1n} y_{1n}	→ coordonatele ultimei stele din prima constelație
.....	
N_m	→ nr. de stele din ultima constelație
x_{m1} y_{m1}	→ coordonatele primei stele din ultima constelație
.....	
x_{mn} y_{mn}	→ coordonatele ultimei stele din ultima constelație

Datele de ieșire se vor scrie în fișierul `STELE.OUT` în următorul format:

- Linia 1: numărul de constelații care conțin Luna;
- Linia 2: se vor afișa în ordine crescătoare numerele de ordine ale constelațiilor care conțin Luna; dacă pe prima linie s-a afișat '0' atunci această linie va fi goală;
- Linia 3: se va afișa `'DA'` dacă există constelația cerută la punctul b) și `'NU'` în caz contrar;
- Linia 4: se va afișa numărul de ordine al constelației găsite la punctul b); dacă o astfel de constelație nu există, linia va fi lăsată goală.

Exemplu:

Pentru fișierul `STELE.IN`:

```
3 9          6 7          11
4            1 9          -4 2
8            3 12         17 6
-1 7         5 12         15 10
-1 4         8 10         3 14
0 11         5 9          -2 12
4 11         9 6          -4 10
-2 9         4 5          12 2
-3 6         4            9 4
1 8          10 6         11 11
4 6          12 8         7 13
9            12 6         11 14
3 3          10 8         (eof)
```

se va genera fișierul `STELE.OUT` cu următorul conținut:

```
3
1 2 4
DA
4
```

Comentariu la exemplu:

Pentru o mai bună înțelegere a problemei voi preciza și care sunt bordurile fiecărei constelații:

Bordura constelației 1:
`(-1,4);(4,6);(4,11);(0,11);(-2,9);(-3,6).`

Bordura constelației 2:
`(3,3);(9,6);(8,10);(5,12);(3,12);(1,9).`

Bordura constelației 3:
`(10,6);(10,8);(12,8);(12,6).`

Bordura constelației 4:
`(-4,2);(12,2);(17,6);(15,10);(11,14);(3,14);(-2,12);(-4,10).`

Timp de implementare: 1 oră și 15 minute

Timp de executare: 2 secunde pe Pentium/133

Observație (care nu ține de testul problemei):
Atenție la complexitate! Trebuie să fie O(m*n*log n).

PR019802: LEMMINGS

Problemă propusă de Cristian Cibu, elev, Liceul Teoretic „Al. Lahovari", Râmnicu Vâlcea.

Enunț:

Niște omuleți foarte simpatici trebuie să scape dintr-un labirint. Ei sunt foarte bine instruiți și știu cum să treacă prin fiecare perete care le iese în față, dar acest lucru îl întârzie pe primul cu o unitate de timp, timp în care ceilalți se apropie de el cu o pătrățică, existând pericolul ca doi lemmingși să se afle pe aceeași pătrățică, ceea ce duce la moartea lor.

Lemmingșii sunt foarte devotați și se sacrifică, rămânând în urmă și arătând celorlalți de fiecare dată când se schimbă direcția de mers. Astfel ei sunt pierduți (deci nu mai pot ieși din labirint), dar nici nu ocupă pătrățica în care se află, așa că ceilalți trec lesne pe lângă el.

Lemmingșii intră pe rând în labirint, prin colțul din stânga jos, mergând de la dreapta la stânga și trebuie să iasă prin colțul opus, mergând tot de la stânga la dreapta.

Ajutați-i voi să se salveze cât mai mulți într-un timp cât mai scurt.

Date de intrare:

Fișierul `LEMMINGS.IN` are structura:

```
nr          → numărul de lemmingși
n           → dimensiunea labirintului (pătratic)
(L sau C)+i a1 b1;a2 b2;....;an bn
            → linie/coloană urmat de nr. ei de ordine,
              coloana/linia de început și de sfârșit între
              care se află peretele.
.....
```

Date de ieșire:

Fișierul `LEMMINGS.OUT` are structura:

```
nr          → numărul de lemmingși salvați
op1         → operația efectuată la primul pas:
                - inainte;
                - spre dreapta;
                - spre stanga;
                - sparge zidul;
op2
...
```

Exemplu:

```
LEMMINGS.IN:                LEMMINGS.OUT :
5                           3
4                           inainte
C1 3 4;0 1;4 5              inainte
C2 0 1;1 2;3 4;4 5          spre stanga
C3 0 1;2 3;4 5              inainte
C4 0 1;2 3;3 4;4 5          inainte
L1 0 1;2 3                  sparge zidul
L2 0 1;1 2;3 4;4 5          inainte
L3 0 1;3 4;4 5              inainte
L4 4 5                      spre dreapta
                            inainte
                            sparge zidul
                            inainte
                            inainte
```

Timp maxim de executare: 5 secunde/test.

PR019803: Triunghi

Problemă propusă de Edison Nica, elev, clasa XII, jud Iași.

Enunț:

Se dă următorul triunghi:

```
1:             1
2:           2   3
3:         4   5   6
4:       7   8   9  10
5:    11  12  13  14 15
6:  16 17 18 19 20 21 22
.......................
```

Să se găsească trei numere care formează un tringhi echilateral a căror sumă este X, astfel încât două numere să se afle pe aceeași linie iar al treilea să se găsească pe o linie cu indice mai mic.

Limite: X aparține intervalului `[1, 1e19]`.

Date de intrare:

X se citește din fișierul `X.IN`.

Date de ieșire:

Cele trei numere se vor scrie (în ordine crescătoare) în fișierul `X.OUT` pe o linie, separate printr-un spațiu. Dacă există mai multe soluții, se va scrie numai una.

Timp de executare: 0.5 secunde pe test.

Exemplu:

```
X.IN                X.OUT
1                   NU
20                  NU
101                 15 41 45
34254               11018 11616 11620
```

P019804: A FI SAU A NU FI... EXCLUS

Problemă propusă de Adrian Manzur, elev, jud. Caraș-Severin.

Enunț:

În 1999, într-o tabără oarecare, într-un oraș oarecare, la un an după incidentele din Poiana Pinului, organizatorii au o mare problemă: cum să-i cazeze pe elevi astfel încât incidentele să nu se repete. Știind că dacă sunt mai mult de doi bucureșteni într-o cameră, va fi dezastru, că fiecare bucureștean are un anumit grad de periculozitate.

Dându-se numărul de paturi dintr-o cameră, ajutați-i pe organizatori să-i cazeze pe elevi.

Datele de intrare:

Fișierul `BUC.IN` conține pe prima linie:

`n` - numărul elevilor;

`p` - numărul paturilor dintr-o cameră;

`b` - numărul bucureștenilor;

iar pe următoarele `b` linii, gradul de periculozitate al fiecăruia.

Se mai știe că numărul elevilor e divizibil cu numărul de paturi dintr-o cameră.

Date de ieșire:

Dacă e posibil să nu fie scandal, fișierul de ieșire `BUC.OUT` va conține mesajul `'Nu exista probleme'`, iar în caz contrar va conține mesajul `'Huston, avem o problema'`, iar pe linia următoare numărul minim de camere devastate.

Timp de executare: MiniM instantaneu

Notă: Camere sunt destule, cu mâncarea mai sunt probleme.

Exemplul 1:

BUC.IN	BUC.OUT
10 2 3	Nu exista probleme
30	
15	
20	

Exemplul 2:

BUC.IN	BUC.OUT
9 3 7	Huston, avem o problema
10	1
20	
3	
18	
5	
7	
35	

INDICAȚII:

1. Încercați backtracking, pentru că sigur se încadrează în timp.
2. Nu vă folosiți de gradul de periculozitate.

P019805: Star Trek

Problemă propusă de Mihai Scorțaru, elev, Liceul de Informatică, Cluj.

Enunț:

Nava spațială *Enterprise* este trimisă pe planeta Alfa Centauri IV pentru a realiza un experiment ciudat. Oamenii de știință au creat (prin inginerie genetică) un nou animal numit Macky, după numele Dr. McCoy (conducătorul grupului de cercetători). Acest animal are posibilitatea de a se deplasa cu viteze foarte mari. Ei leagă de urechea animalului un clopoțel care sună la fiecare secundă. De fiecare dată când aude clopoțelul, animalul se sperie și își dublează viteza. Știind că la început animalul pornește cu 1 m/s, savanții vor să calculeze timpul în care animalul parcurge o anumită distanță (dublarea vitezei are loc instantaneu). Cum domnul Sulu este bolnav, Cpt. James T. Kirk apelează la dumneavoastră pentru a calcula perioada de timp.

Date de intrare:

Fișier `INPUT.TXT` conține distanța `d` (în metri).

`1<=d<=2.000.000.000`

Date de ieșire:

Fișierul `OUTPUT.TXT` va conține perioada `t` (cu o precizie de 1 miime de secundă).

Timp de executare: 2,457 secunde/test.

P019806: Multiplu

Problemă propusă de Andrei Vancea, elev, Liceul de Informatică, Cluj.

Enunț:

Se dau numerele naturale n ($2 \le n \le 15000$) și b ($2 \le b \le 9$). Să se calculeze un multiplu al lui n format doar din cifre de 0 și b.

Date de intrare: fișierul `MULTIPLU.IN` va conține n și b.

Date de ieșire: fișierul `MULTIPLU.OUT` va conține multiplul determinat de program.

Exemplu:

`MULTIPLU.IN:` 40 3

`MULTIPLU.OUT:` 3000

Timp maxim de executare: 1 secundă.

P019807: Scufița Roșie

Problemă propusă de Badiu Nicolae, elev, Liceul „B.P.Hașdeu", clasa a XI-a.

Enunț:

În fiecare duminică, Scufița Roșie trebuie să plece de acasă ca să o viziteze pe bunicuța ei. Însă pentru a ajunge la ea, Scufița Roșie trebuie să meargă pe un drum foarte bine ales. Deoarece casa se află în pădure, fetița a inventat un dispozitiv ingenios cu care poate să sară peste orice obstacol cu o deviație constantă pe OX și OY. Astfel, ea poate sări peste gropi, copaci și chiar să evite eventualii lupi care o pândesc.

Având la dispoziție un laptop performant și harta pădurii digitizată, ea realizează un program care să o ajute să determine un drum sigur către bunicuța ei. Dacă găsește măcar un drum corespunzător, ea îl va alege pe cel mai scurt. Dacă nu, va aștepta până duminica viitoare pentru ca să primească prin satelit o hartă ce permite soluții.

Date de intrare: Numele fișierului de intrare se citește din primul parametru al programului.

Structura fișierului de intrare:

- pe prima linie se află numărul de linii și coloane pentru matricea pătratică a hărții pădurii ($1 \le n \le 600$);
- pe următoarele n linii se vor afla câte n cifre despărțite prin spațiu, reprezentând:
  - 0 - drum fără primejdie
  - 1 - copac
  - 2 - groapă
  - 3..9 - lup

Observație: Oricare cifră nenulă este un obstacol pe care Scufița Roșie nu poate să cadă în urma unei sărituri.

- pe următoarea linie se află linia și coloana în plan a casei bunicuței ($0 \le x, y \le n$).
- pe ultima linie se găsesc două numere ce reprezintă deviația pe OX și pe OY a dispozitivului care o aduce pe Scufița Roșie într-o altă poziție ($0 \le L1, L2 \le n$).

Observație: Scufița Roșie poate să sară numai în interiorul hărții și numai cu o incrementare sau decrementare în același timp pe OX cât și pe OY, folosindu-se și L1 și L2 pe oricare din axe.

Scufița Roșie pornește din coordonata (1, 1) a hărții.

Restricții: Nu se poate incrementa sau decrementa în săritura Scufiței Roșii pe ambele axe numai L1 sau L2.

Date de ieșire: Fișierul de ieșire va fi OUTPUT.TXT, având următoarea structură:

- pe prima linie se va afișa numărul minim de sărituri pe care le face Scufița Roșie până la casa bunicii, dacă există măcar o soluție în condițiile date. În caz contrar se va afișa mesajul 'NU ESTE POSIBIL'.

Exemple:

Intrare:
```
5
0 0 0 0.0
1 0 0 1 0
0 1 0 0 0
1 0 0 1 1
0 0 1 0 0
5 4
1 2
```
OUTPUT.TXT:
```
3
```

Intrare:
```
6
0 0 0 0 0 0
1 0 0 1 0 0
0 1 0 0 0 0
1 0 0 1 1 0
0 0 1 0 0 0
0 0 0 0 0 0
6 3
1 3
```
OUTPUT.TXT:
```
NU ESTE POSIBIL
```

Observație: Nu se fac verificări la citire sau scriere.

Timp maxim de executare:

8 secunde/test pentru un calculator 586/133Mhz.

P019808: Giga Vodă

Problemă propusă de prof. Emanuela Mateescu, Liceul de Informatică, Iași.

Enunț:

În Țara lui *Giga Vodă* pregătirile sunt în toi: preafrumoasa-i fiică, *Octeta*, se va mărita cu cel mai luminat pretendent la dalba'i mână.

La poarta împărăției au dat buluc o mândră ceată de voinici, dornici să câștige mâna (și averea) Octetei.

Acela care va reuși într-un timp cât mai scurt să rezolve problema dată de Giga Vodă, va primi în plus și jumătate din împărăție.

Amar de cel ce nu va izbuti: va fi trimis acasă!!!

Problema constă în găsirea celei mai lungi înșiruiri de vorbe, luate dintr-un hrisov, care are cel mult 500 de vorbe, fiecare vorbă având cel mult 20 slove.

Numele hrisovului este dat de Vodă, în el apărând pe fiecare rând câte o vorbă.

O vorbă din înșiruire este imediat urmată de o altă vorbă numai dacă a doua se obține din prima, aplicând una din următoarele transformări:

- înlocuirea unei slove;
- inserarea unei slove;
- ștergerea unei slove.

Voinicii puteau să-și aleagă vorbele numai în ordinea în care ele apăreau în hrisov.

După maxim 1 secundă grămăticul Logo trebuie să primească un hrisov cu numele OUTPUT.TXT care are pe primul rând un număr L, ce arată lungimea șirului de vorbe obținut din hrisovul dat de Vodă.

Pe următoarele L rânduri sunt condeiate vorbele găsite de voinici.

Ca să-i ajute, Vodă l-a pus pe Logo să le dea o pildă.

Hrisovul lui Vodă	*Hrisovul voinicului* (OUTPUT.TXT)
pin	6
pic	pin
spic	pina
pina	pana
pana	spana
spil	span
spana	spin
span	
pom	
spin	

P019809: Amnistie

Problemă propusă de prof. Emanuela Mateescu, Liceul de Informatică, Iași.

Enunț:

Executând prevederile unei amnistii parțiale, un gardian trebuie să elibereze din cele N ($1 < N < 2.0E9$) celule ale închisorii cel mult MAX ($1 < MAX < N$) deținuți. Celulele sunt așezate în rând, sunt numerotate de la 1 la N și fiecare celulă este ocupată de un singur deținut. Regulamentul de amnistiere trebuie respectat întocmai:

Gardianul trebuie să deschidă toate celulele închisorii.

La pasul următor, gardianul trebuie să închidă fiecare a doua celulă.

La pasul 3, gardianul trebuie să ia celulele din 3 în 3, răsucind cheia în broască (cele deschise vor fi închise, iar cele închise deschise). Aceste operații trebuie să se repete, astfel încât la pasul `i` gardianul va lua celulele din `i` în `i` și va răsuci cheia în broasca lor. Numărătoarea trebuie să înceapă întotdeauna din dreptul primei celule.

Deținuții ale căror celule au rămas deschise după efectuarea tuturor acestor operații vor fi eliberați dacă numărul lor nu depășește `MAX`, numărul maxim de amnistieri permise. În caz contrar, procedeul se reia, gardianul luând în considerare la numărătoare în continuare numai celulele deținuților selectați la pasul precedent, până când numărul de deținuți selectați în urma aplicării complete a procedeului nu depășește `MAX`.

Cunoscând `N` - numărul de celule din închisoare, și `MAX` - numărul maxim de amnistieri permise, determinați celulele norocoase.

Date de intrare: Fișierul de intrare, al cărui nume se citește de la tastatură, conține pe prima linie `N`, numărul de celule din închisoare, iar pe a doua linie `MAX`, numărul maxim de amnistieri permise.

Date de ieșire: În fișierul de ieșire `OUTPUT.TXT` vor fi afișate pe linii separate numerele celulelor din care vor fi eliberați deținuții, în ordinea crescătoare a numerelor lor.

Exemplu:

Pentru fișierul de intrare

```
20
3
```

fișierul de ieșire `OUTPUT.TXT` va conține:

```
1
16
```

Timp maxim de executare: 0.5 secunde/test.

P019810: Înmulțiri

Problemă propusă de prof. Adrian Niță, Liceul „Emanoil Gojdu", Oradea.

Enunț:

Gigel, elev olimpic, a fost recompensat pentru rezultatele deosebite la informatică cu o tabără la *Poiana Lu' Pin*. În prima zi, neavând program clar, a făcut o excursie până la Mănăstirea *Lu Cio Lan*.

Aici probleme mari: părintele chelar era bolnav!!! Avea *longint*-ită acută, manifestată prin alergie la numere mari.

Gigel, săritor din fire, s-a oferit să-l ajute, folosind un calculator de buzunar, care s-a dovedit inutilizabil pentru numerele mari necesare preasfinției sale.

Cum mănăstirea primise ca donație un *La' p(o)top* (*Pent' um Contra*), Gigel s-a oferit să-i facă părintelui un program de înmulțiri pentru numere întregi pozitive cu cel mult 20 de cifre.

Părintele era pretențios: dorea înmulțirile afișate pe ecran, așa cum învățase în clasa a III-a primară.

Exemplu:

```
  123
   25
  ---
  615
 246
 ----
 3075
```

Numerele se citesc de la tastatură, separate prin spațiu. Afișarea rezultatului, pe ecran, se face instantaneu.

P019811: Amnistie

Problemă propusă de prof. Dana Lica, Liceul „I. L. Caragiale", Ploiești.

Enunț:

Cartierul General al invadatorilor se află în orașul `p`, stabilit inițial.

Cunoscându-se rețeaua de drumuri cu sens unic, ce leagă orașele din Țara Fâșiilor și costurile deplasării blindatelor pe acestea, să se găsească orașul care poate fi invadat (plecând din orașul `p`) pe un număr maxim de trasee (drumuri), toate de cost minim.

Date de intrare: Numele fișierului de intrare se citește de la tastatură. Acesta va conține datele sub forma următoare:

- Pe prima linie sunt scrise două numere întregi: `n` și `p` (separate prin spațiu), reprezentând numărul de orașe (`n<=100`) și respectiv orașul de plecare.
- Pe următoarele linii sunt scrise triplete de numere: `i`, `j`, și `c` (separate prin spațiu), reprezentând câte un drum direct de la `i` la `j` de cost `c`.

Date de ieșire: Ieșirea va fi în fișierul `OUT.TXT` și va conține:

- Pe prima linie, tripletul de numere: `nr`, `s`, `min` (separate prin spațiu), reprezentând numărul maxim (`nr`) de trasee existente între `p` și `s`, toate de același cost minim (`min`).
- Pe următoarele `nr` linii vor fi scrise traseele, pentru fiecare specificându-se, în ordine, orașele străbătute.

Observație: În cazul în care există mai multe soluții, se va afișa una singură

Exemplu:

Pentru fișierul de intrare:

```
7 1
1 2 1
2 3 5
2 6 1
5 3 2
6 5 4
6 7 3
7 4 1
7 3 1
3 4 2
5 1 1
```

Fișierul de ieșire va conține:

```
2 3 6
1 2 3
1 2 6 7 3
```

Timp de executare: 1 secundă/test.

P019812: Polinom

Problemă propusă de prof. Anastasiu Doru Popescu, Liceul „Radu Greceanu", Slatina.

Enunț:

Se consideră n număr natural, $n \leq 10000$ și $a_0, a_1, \ldots, a_n$ un șir de numere naturale mai mici sau egale cu 10000.

Notăm cu $P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ polinomul cu coeficienții dați de termenii șirului.

Pentru x dat, de tip real, determinați:

a) o rearanjare a componentelor șirului astfel încât valoarea $P(x)$ a polinomului să fie minimă;

b) valoarea minimă obținută la punctul a).

Datele de intrare se citesc din fișierul text `POL.IN`, având structura:

x
n
a_0
a_1
...
a_n

Datele de ieșire se vor scrie în fișierul text `POL.OUT` având structura:

a_0
a_1
... elementele șirului în ordinea care realizează minimul de la punctul a)
a_n
min valoarea minimă cerută la punctul b)

Exemplu:

`POL.IN:`

```
3
2
1
2
1
```

`POL.OUT:`

```
2
1
1
14
```

Timp maxim de executare: 2 secunde pentru fiecare test.

P019813: Poșta

Problemă propusă de prof. Emanuela Mateescu, Liceul de Informatică, Iași.

Enunț:

Un poștaș trebuie să treacă zilnic pe toate străzile din cartierul său. În fiecare zi el pleacă de la poștă și trebuie să se întoarcă la poștă. Cum în cartier sunt `m` străzi ($1 < m < 5000$), după o săptămână deja își pune problema să își optimizeze traseul. Pentru aceasta a început prin a numerota intersecțiile, începând cu 1 pentru intersecția în care se găsește poșta. Pentru a se feri de complicații, poștașul a considerat că într-o intersecție nu trebuie neapărat să se întâlnească două străzi, este suficient ca locul să fie un capăt de stradă. Apoi și-a făcut o hartă, pe care a marcat toate intersecțiile și toate străzile. A observat că în cartierul său pot exista străzi distincte care au ambele capete respectiv în aceleași intersecții și că de la poștă poate ajunge pe orice stradă dorește, trecând doar pe străzi din cartierul său. Poștașul ar prefera un traseu pe care să nu fie nevoit să treacă de două ori pe o aceeași stradă.

Scrieți un program care să testeze dacă poștașul are șanse să își îndeplinească visul și dacă da, să determine un traseu convenabil.

Datele de intrare se citesc dintr-un fișier al cărui nume se citește de la tastatură.

Pe prima linie se găsește `m`, numărul de străzi din cartier.

Pe fiecare din următoarele `m` linii se găsesc câte două numere separate prin spațiu, care reprezintă numerele asociate intersecțiilor ce reprezintă capetele străzilor din cartier.

Date de ieșire: Rezultatele vor fi afișate în fișierul `OUTPUT.TXT`.

Prima linie din fișier conține mesajul `'DA'`, dacă problema are soluție, respectiv `'NU'`, în caz contrar.

Dacă problema are soluție, în fișierul de ieșire vor fi afișate intersecțiile în ordinea în care sunt parcurse pe un traseu optim (unul în care fiecare stradă este parcursă o singură dată), începând cu 1, intersecția de plecare și terminând, de asemenea, cu 1, fiecare intersecție pe o linie separată.

Exemple:

Intrare:	OUTPUT.TXT
5	DA
1 4	1
1 2	4
1 3	1
1 4	2
2 3	3
	1

Intrare:	OUTPUT.TXT
1 2	NU
1 4	

Intrare:	OUTPUT.TXT
1	DA
1 1	1
	1

Timp maxim de executare: 1 secundă/test.

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.

Această carte este concepută în conformitate cu Programa analitică a clasei a XI-a, profilul Informatică. Ea poate fi considerată o introducere elementară în limbajul C. Fiecare noțiune este explicată în mod clar și utilizată în programe simple. De asemenea, capitolele sunt însoțite de exerciții propuse care contribuie la fixarea și înțelegerea noțiunilor întâlnite în capitolele respective.

Cartea se adresează atât elevilor cât și celor interesați să se inițieze în limbajul C care în prezent este unul dintre cele mai utilizate limbaje de programare.

Liviu Negrescu este cercetător principal gr. I la societatea SOFTWARE ITC S.A., filiala Cluj, unde se ocupă cu dezvoltări de programe de sistem și aplicative.
În același timp, desfășoară o activitate didactică prestigioasă de peste 15 ani, la Facultatea de Calculatoare de la Universitatea Tehnică din Cluj.
În decursul anilor a fost titular a mai multe cursuri, ca de exemplu:

- Limbaje de programare și compilatoare;
- Proiectarea translatoarelor etc.

Liviu Negrescu are o serie de lucrări științifice publicate în diferite reviste de specialitate. În colaborare cu alți cercetători de la SOFTWARE ITC S.A. a implementat sisteme cu aplicații în diferite domenii:

- Construcția de compilatoare;
- Interfețe utilizator;
- Prelucrări de date din domenii economice și medicale.

Liviu Negrescu este autorul mai multor cărți apărute în decursul anilor, cu privire la diferite limbaje de programare (CPL1-subset al limbajului PL/1, CDL-limbaj de descriere al compilatoarelor precum și referitoare la limbajele C și C++. Lucrarea beneficiază de experiența vastă a autorului câștigată atât în activitatea de cercetare cât și în cea didactică.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

STRUCTURI înlănțuite în limbajul JAVA

de Horia Georgescu

Java este mai mult decât un nou limbaj. Este o nouă abordare a informaticii. Speranțele care se pun în el sunt imense și doar viitorul ne poate spune dacă va deveni o *lingua franca* a computerelor secolului XXI. Indiferent ce se va întâmpla, cititorii revistei noastre nu vor fi luați prin surprindere.

Ne propunem ca prin articolul de față să începem o serie de articole dedicate limbajului Java, care pare să fie „limbajul de sfârșit de mileniu".

Cu ce să începem? Pentru a face reclamă limbajului ar trebui să începem cu partea cea mai spectaculoasă a sa și anume applet-urile. Am ales însă altă cale din următoarele motive:

- limbajul Java oferă deschideri importante pentru dezvoltarea de soft în mult mai mare măsură decât applet-urile;
- este natural să ne întrebăm dacă ceea ce se poate realiza în limbajele Pascal, C și C++ poate fi realizat și în Java;
- dorim să continuăm tradiția (putem vorbi despre acest lucru după 7 ani) Gazetei de Informatică, prin care se acordă o atenție specială scrierii de algoritmi.

De aceea ne propunem pentru început să răspundem la următoarea întrebare: *Dacă se afirmă despre limbajul Java că elimină pointerii, atunci cum pot fi implementate în acest limbaj liste, stive, cozi, arbori etc.?*

Introducere

Limbajul Java a apărut în urmă cu doi ani. Drept părinte al său este unanim recunoscut *James Gosling*. Acest răstimp a fost însă suficient ca Java să se impună și să capete o largă recunoaștere și utilizare.

Înainte de a face o scurtă trecere în revistă a facilităților oferite de Java, atenționăm pe cititor că poate sări pe moment această parte a articolului, dar este indicat să o parcurgă măcar după ce a citit restul articolului.

De asemenea atragem atenția că, la fel ca în cazul celor mai multe produse soft, prezentarea „secvențială" a limbajului nu este posibilă. De aceea, în mod inerent, la început multe aspecte vor fi prezentate incomplet, scopul fiind ca cititorul să ajungă cât mai repede în situația de a scrie, compila și executa programe (numite în Java *aplicații*).

Iată pe scurt avantajele utilizării limbajului Java, așa cum au fost ele evidențiate de Sun Microsystems.

- ***Simplitate.*** Deși împrumută multe aspecte din limbajele C și C++, limbajul Java renunță la multe trăsături neesențiale ale acestora ca de exemplu structuri, supraîncărcarea operatorilor, moștenire multiplă. Aparent, limbajul are mai puține grade de libertate decât C și C++, dar în realitate poate realiza aceleași lucruri și anume cu un plus de claritate.
- ***Orientarea spre obiecte*** *(object-oriented).* Ca și C++, Java folosește clase pentru organizarea logică a codului. La executare, programul creează *obiecte* de tipul claselor, care pot fi gândite ca niște călăuze care execută ele funcțiile clasei. Este admisă și larg utilizată moștenirea claselor, dar nu este permisă moștenirea multiplă (ea este suplinită de o altă facilitate, numită *interfață*).
- ***Tipizare statică.*** Obiectele folosite într-un program Java trebuie obligatoriu declarate înainte de a fi folosite; în acest mod se ușurează sarcina compilatorului de a detecta conflicte de tip.
- ***Compilare prealabilă urmată de executare pe mașina gazdă.*** Înainte de a fi executat, un program scris în Java trebuie compilat. Compilatorul produce un fișier „byte-code" (o secvență de instrucțiuni ale unei *mașini virtuale Java*), care poate apoi să fie executat sub orice sistem de operare ce include un interpretor Java. În acest mod este asigurată portabilitatea produselor scrise în Java.
- ***Posibilitatea lansării mai multor fire de executare.*** Limbajul permite lansarea mai multor fire de executare, care asigură executarea concurentă a mai multor acțiuni. De exemplu, este posibil să se afișeze o imagine pe ecran, concomitent cu citirea unor date de intrare.

Orice aplicație are cel puțin un fir de executare. Exemplul de mai sus este foarte puțin sugestiv, deoarece facilitatea analizată permite simularea executării de programe paralele și concurente, inclusiv o gama largă de sincronizări. În momentul în care interpretorul detectează mai multe procesoare pe mașina gazdă, acestea sunt efectiv folosite pentru executarea concurentă a mai multor acțiuni.

- *Colectorul de gunoaie.* Java permite crearea explicită de obiecte (de tipul unei clase), dar nu include posibilitatea de a distruge aceste obiecte. Această sarcină este preluată de *colectorul de gunoaie*, care marchează obiectele ce nu mai sunt folosite și eliberează spațiul ocupat de ele; eliberarea nu se face neapărat imediat, ci atunci când spațiul disponibil curent nu mai poate satisface o nouă cerere.
- *Securitate ridicată.* Java asigură un grad de securitate ridicat, uneori cu un plus de efort din partea programatorului. Accesul la memorie este posibil numai prin verificarea prealabilă a drepturilor de acces. Lipsa pointerilor face ca accesarea unor zone de memorie pentru care accesul nu este autorizat să nu fie posibilă. De asemenea, este asigurată nepătrunderea virușilor pe timpul executării unei aplicații.
- *Extensibilitate.* Limbajul Java permite includerea de *metode native*, adică de funcții scrise în alt limbaj (de obicei C++). Metodele native sunt legate dinamic la programul Java la momentul executării, rolul lor fiind în principal de a mări viteza de executare pentru anumite secvențe din program.
- *Claritate.* Așa cum s-a menționat, eliminarea unor facilități neesențiale din C și C++ conduce la programe mai ușor de înțeles. În plus, se consideră că limbajul Java este ușor de înțeles, cu condiția ca cel care îl prezintă să fi înțeles el însuși foarte bine mecanismele puse la dispoziție (ușor de spus, greu de realizat!).
- *Applet-uri.* Limbajul Java permite programatorului de a crea un tip special de programe, numite *applet-uri*, care pot fi incluse în paginile Web. Când utilizatorul vizualizează o pagină Web ce include un applet, mașina la care este conectat transmite applet-ul utilizatorului iar, navigatorul utilizatorului execută applet-ul (se presupune că mașina pe care lucrează utilizatorul are un interpretor Java). Ne mărginim la a menționa că applet-urile pot include grafică, animație, sunet, jocuri etc.

Un prim program scris în Java

Nu vom căuta să fim originali, ci vom prezenta un prim program „clasic":

```
class unu {
  public static void main(String[] sir) {
    for (int i=0; i<sir.length; i++)
      System.out.println(sir[i]);
  }
}
```

Deși scurt, programul necesită o serie de explicații.

În Java, literele mici și literele mari sunt tratate diferit; de exemplu `unu` și `Unu` sunt doi identificatori diferiți.

Programul de mai sus este compus dintr-o singură clasă și anume `unu`. Programul sursă se va numi `unu.java`.

Clasa `unu` conține doar o funcție (numită în Java *metodă*), cu numele `main`. Aceasta este metoda „principală": la executarea programului, sistemul detectează și execută metoda `main` a clasei al cărei nume coincide cu numele programului.

Metoda `main` trebuie declarată cu atributele (numite în Java *modificatori*) `public`, `static` și `void`. Ne mărginim la a preciza semnificația lui `void`: metoda nu întoarce vreo valoare. În metoda `main` apare în mod obligatoriu un argument de tip tablou de șiruri de caractere. Cum tablourile sunt indexate în Java începând cu `0` și cum lungimea tabloului `sir` poate fi referită prin `sir.length`, instrucțiunea `for` tipărește, cu ajutorul metodei `println`, la ieșirea standard (identificată prin `System.out`) pentru fiecare `i` luând valori în intervalul `0..sir.length-1` conținutul elementului `sir[i]`.

Reamintim că numele programului sursă este `unu.java`. Cum extensia este formată din 4 caractere, se recomandă să *nu* folosim pentru editare, copiere etc. produsul Norton Commander, care lucrează (în versiuni mai vechi) cu nume de fișiere în care extensia este formată din 3 caractere; de altfel, în situația în care lucrăm sub DOS, este totdeauna preferabil să lucrăm cu comenzi DOS și nu cu „ajutoare" de tipul Norton Commander.

Compilarea se face folosind compilatorul `javac.exe` astfel:

```
javac unu.java
```

În urma compilării ia naștere fișierul `unu.class`. Pentru acest fișier putem comanda executarea sa apelând interpretorul `java.exe` astfel:

```
java unu lista de identificatori
```

De exemplu, pentru programul de mai sus, după ce s-a trecut de faza de compilare, dacă dăm comanda:

```
java unu Unu Doi Trei
```

vectorul `sir` va fi format din trei componente, iar la ieșire va apărea:

```
Unu
Doi
Trei
```

Modul de compilare și executare a programelor Java descris mai sus este cel mai simplu posibil. El poate fi folosit utilizând *Java Development Kit*. Menționăm însă că în prezent există medii de programare în Java, ca de exemplu, *Java Café* (Symantec) și *Java Visual Age* (IBM).

Clase, câmpuri și metode

Clasa este unitatea de programare fundamentală în Java. O clasă nu poate conține altă clasă. Orice clasă este formată din *câmpuri*, *metode* și *constructori*. Câmpurile și metodele

unei clase formează împreună *membrii* acelei clase.

Câmpurile unei clase sunt variabile atașate clasei respective. *Metodele* sunt funcțiile declarate în interiorul clasei. O metodă nu poate conține altă metodă. Numele funcției este prefixat cu tipul rezultatului întors de funcție (`void` dacă metoda nu întoarce vreo valoare) și urmat de lista parametrilor, cuprinsă între paranteze rotunde; cele două paranteze rotunde trebuie specificate chiar dacă nu există parametri. Numele metodei împreună cu lista tipurilor parametrilor, în ordinea lor de apariție, formează *signatura* metodei. În aceeași clasă pot apare mai multe metode cu același nume, dar cu signaturi diferite.

La apelarea oricărei funcții (numită în Java *invocare*) transmiterea parametrilor se face *prin valoare*.

Clasele sunt considerate tipuri. Entitățile al căror tip este o clasă se numesc *obiecte* ale clasei respective. Se mai spune că obiectele sunt *instanțieri* (*instanțe*) ale claselor.

Să considerăm următoarea clasă, care nu conține câmpuri ci numai două metode numite `met` și `scrie`:

```
class Clasa {
  public boolean met(int x) {
    return x==0; /* == este operatorul de comparare */
  }
  public void scrie(String s) {
    System.out.println(s);
  }
}
```

unde un șir de caractere cuprins între `/*` și `*/` reprezintă un comentariu.

Crearea unui obiect `Ob` al acestei clase (având drept tip această clasă) se face prin:

```
clasa Ob = new clasa();
```

unde repetarea tipului obiectului (`clasa clasa`) va fi explicată mai târziu.

Având acum la dispoziție acest obiect, avem acces la câmpurile (aici inexistente) și la metodele clasei prin prefixare (calificare) cu numele obiectului. Astfel pentru scrierea unei variabile `sir` de tip `String` (șir de caractere), vom invoca metoda `scrie` astfel:

```
Ob.scrie(sir);
```

Un obiect poate fi interpretat ca o persoană care dispune de informații (câmpurile clasei) și de modalități de prelucrare a lor (metode); o metodă prelucrează atât câmpurile obiectului care o invocă, precum și parametrii care îi sunt transmiși la invocare. Crearea unui obiect revine la a face apel la o persoană care să preia rolul de a executa metodele clasei. Invocarea unei metode pentru un obiect revine la a pune acel obiect (persoană) să execute codul metodei respective.

Modificatorul `public` stabilește că accesul la metodele clasei se poate face de oriunde unde clasa este accesibilă. Menționăm că poate fi folosit și modificatorul `private`, care restrânge accesul la interiorul clasei; în acest mod se preîntâmpină un acces dintr-un loc nepotrivit. Aceiași modificatori pot fi folosiți și pentru câmpurile claselor.

Să ne întoarcem la declararea:

```
clasa Ob = new clasa();
```

Pare ciudată inserarea parantezelor rotunde deschisă și închisă, construcție tipică pentru un apel sau o declarare de metodă cu nici un parametru. Lucrurile nu mai par ciudate dacă precizăm că la crearea unui obiect cu `new` are loc un apel implicit al unui *constructor* al clasei respective.

Un *constructor* seamănă cu o metodă prin modul de invocare. Un constructor nu este însă o metodă deoarece nu întoarce nici un rezultat; scopul său este de a inițializa unele dintre câmpurile clasei de al cărei tip este obiectul nou creat. Constructorul are același nume ca și clasa respectivă. La fel ca pentru metode, într-o clasă pot exista mai mulți constructori, cu condiția ca signaturile lor să fie diferite. Prin lipsă (ca mai sus) este considerat constructorul cu nici un argument, al cărui efect este nul.

În limbajul Java nu există înregistrări. Ele pot fi simulate folosind clase (eventual fără metode). Acest lucru este ilustrat de următorul program, care ilustrează cum o metodă poate întoarce mai multe rezultate (reamintim că transferul parametrilor are loc numai prin valoare):

```
class Record {
  public int i; public boolean b;
  Record(int ii, boolean bb) {
        /* constructorul clasei */
    i = ii; b = bb;
        /* este operatorul de atribuire */
  }
}

class C {
  public Record metoda (int i) {
    Record R = new Record(i+1,i<0);
    return R;
  }
}

class Inreg {
  public static void main(String[] sir) {
  int i; Record R; String s;
  i = -2; C Ob = new C(); R = Ob.metoda(i);
  if (R.b) s = "true";
  else s = "false";
  System.out.println(R.i+s);
  }
}
```

Exemplul este forțat, deoarece filosofia programării orientate spre obiecte nu concepe gruparea unor câmpuri fără a specifica și operațiile ce urmează a fi executate asupra lor.

Crearea unei liste înlănțuite

Vom prezenta în acest paragraf o primă modalitate de a crea și prelucra o listă înlănțuită.

Așa cum am anunțat în primul paragraf al articolului, limbajul Java asigură o securitate ridicată, care însă presupune un mic efort suplimentar la redactarea programului. Concret, chiar o citire simplă de la dispozitivul standard de intrare trebuie însoțită de măsuri de precauție.

Limbajul Java pune la dispoziția utilizatorului multiple clase predefinite, grupate în *pachete* (asemănătoare bibliotecilor). Un pachet pentru operațiile de intrare/ieșire este pachetul `java.io`. Pentru a avea acces la clasele și deci și la metodele acestor clase va trebui să anunțăm explicit că dorim acest acces, ceea ce se realizează simplu prin:

```
import java.io.*;
```

În programul care urmează vom utiliza metoda `read`, care nu are nici un parametru, a clasei `InputStream` din pachetul `java.io`. Ca urmare a invocării `System.in.read()` este citit și întors un byte de la intrare, dacă nu s-a ajuns la sfârșit de fișier. Valoarea întoarsă este un întreg din intervalul 0..255. Interpretarea acestei valori ca fiind caracterul cu numărul de ordine respectiv ne conduce la obținerea caracterului care a fost citit. Interpretarea menționată se realizează printr-o conversie la tipul `char`, care la rândul ei se realizează prin prefixarea apelului cu `(char)`. Rezumând, dacă `ch` este o variabilă de tip caracter, atunci prin:

```
ch = (char) System.in.read();
```

se obține în `ch` următorul caracter din fluxul de intrare, dacă există un astfel de caracter. Această instrucțiune trebuie plasată în contextul:

```
try {
  ..; ch = (char) System.in.read(); ..
}
catch (IOException e) {
   System.out.println("I/O Error")
}
```

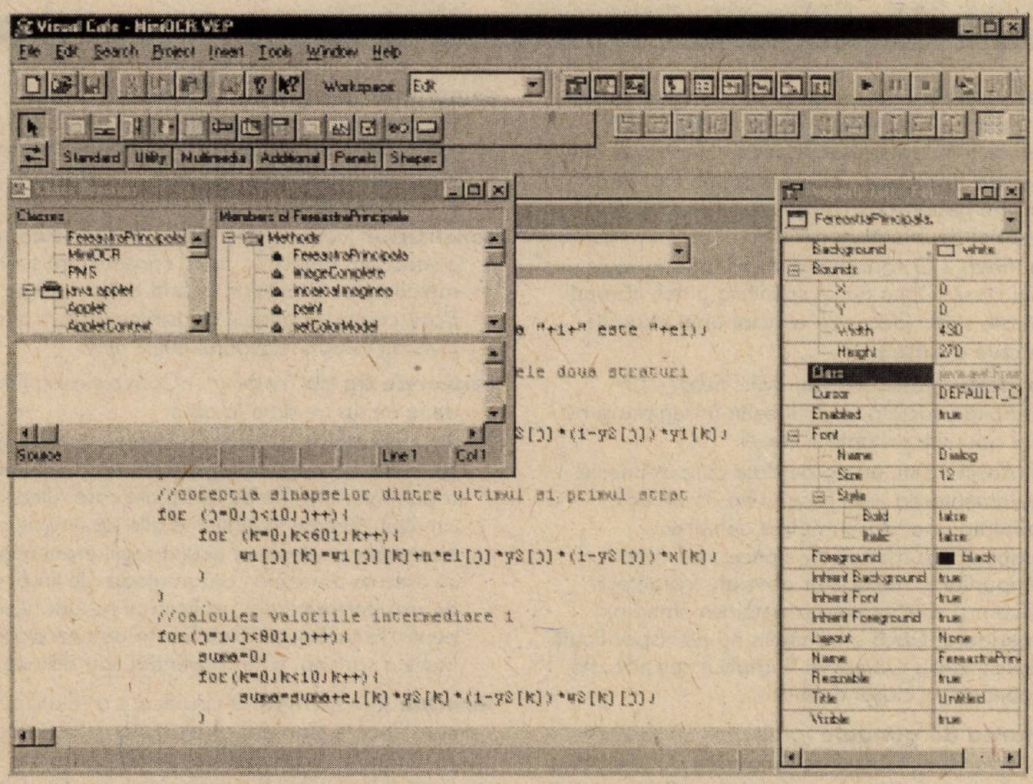
pentru a spune explicit că în cazul când intervine o eroare la citire este tipărit mesajul specificat.

După cum știm, ideea de bază este de a forma „celule" care să conțină un caracter și „adresa" următoarei celule. Pentru aceasta vom defini o clasă `elem` formată din:

- câmpurile `c` (de tip caracter) și `leg` (de tip `elem`);
- metodele `parc_dir` și `parc_inv` ce au un parametru `x` de tip `elem` și întorc un rezultat de tip șir de caractere ce corespunde parcurgerii în ordine normală (respectiv în ordine inversă) a șirului de caractere ce începe de la „adresa" `x`.

În clasa `lista`, ce conține metoda principală, variabilele `p` și `u` vor memora „adresele" primului și ultimului element din listă.

Programul corespunzător este prezentat în continuare.



Mediul de dezvoltare Visual Cafe de la Symantec. Elementele principale: truse de unelte; ferestre de editare; navigatoare prin clasele aplicației; ferestre pentru proprietățile claselor sau elementelor din interfața utilizator a viitoarei aplicații.

```
import java.io.*;

class elem {
  char c; elem leg;

  elem(char ch) {
    c=ch; leg=null;
  }

  elem adaug(elem u, char ch){
    elem x = new elem (ch); u.leg = x;
    return x;
  }

  String parc_dir(elem x) {
    if (x==null) return("");
    else return( x.c + parc_dir(x.leg));
  }

  String parc_inv(elem x) {
    if (x==null) return("");
    else return( parc_inv(x.leg) + x.c);
  }
}
```

```
class lista {
  public static void main(String arg[]) {
    elem p=null, u; char ch;
    try {
      ch = (char) System.in.read();
      p = new elem(ch); u = p;
      ch = (char) System.in.read();
      while (ch != '$') {
            /* != semnifică "diferit" */
        u = u.adaug(u,ch);
        ch = (char) System.in.read(); };
    }
    catch (IOException e)
      { System.out.println("Input Error"); }
    System.out.println("********************");
    System.out.println(p.parc_dir(p));
    System.out.println(p.parc_inv(p));
  }
}
```

Să analizăm programul de mai sus.

Clasa `elem` conține:

- câmpurile `c` de tip `char` și `leg` de tip `elem`;
- un constructor, care primește un caracter; la crearea unui nou obiect de tip `elem`, obiectul va primi în câmpul `c`

Glosar de termeni Java

applet (*applet*) – Un program scris în Java pentru a rula într-un browser Web compatibil Java, cum ar fi HotJava sau Netscape Navigator.

byte-code (*bytecode*) – Cod independent de mașină generat de compilatorul Java și executat de interpretorul Java.

câmp (*field*) – O variabilă definită în interiorul unei clase. Dacă nu se specifică altfel, câmpul nu este static (deci este asociat unei anumite instanțe a clasei).

câmp static (*class field* sau *static field*) – O variabilă asociată unei clase în întregime și nu unei anumite instanțe a clasei.

clasă (*class*) – Este un tip de date care definește implementarea unui anumit tip de obiecte. Definirea unei clase implică definirea câmpurilor și metodelor statice, definirea câmpurilor și metodelor asociate instanțelor precum și specificarea superclasei imediat superioare. Dacă superclasa nu este specificată explicit, clasa *Object* va fi implicit considerată superclasă a clasei definite.

colectarea de gunoaie (*garbage collection*) – Detectarea și eliberarea automată a zonelor de memorie care nu mai sunt folosite. Sistemul de rulare Java realizează colectarea gunoaielor, așa încât programatorii nu eliberează niciodată obiectele în mod explicit.

compilator (*compiler*) – Un program care translatează codul sursă în cod care poate fi executat de un calculator. Compilatorul Java translatează codul sursă Java în *byte-code*. Vezi și „interpretor".

constructor (*constructor*) – O pseudo-metodă care creează un obiect. În Java, constructorii sunt metode ale instanțelor, având același nume cu clasa căreia îi aparțin. Constructorii Java sunt invocați folosind cuvântul-cheie *new*.

conversie de tip (*casting*) – Conversia explicită de la un tip de date la altul.

fir de executare (*thread*) – Unitatea fundamentală de executare a unui program. Un proces poate avea mai multe fire de executare care rulează simultan, fiecare îndeplinind alte sarcini, de pildă să aștepte producerea unui anumit eveniment sau să execute o sarcină consumatoare de timp, dar de rezultatele căreia programul nu are nevoie pentru a continua. Când un fir de executare își încheie sarcina, este suspendat sau distrus.

ierarhie (*hierarchy*) – O clasificare a relațiilor în care fiecare element cu excepția celui mai de sus (numit „rădăcină" - *root*) este o formă specializată a celui de deasupra sa. Fiecare element al ierarhiei poate avea unul sau mai multe elemente sub el. În ierarhia claselor Java, rădăcina este clasa *Object*.

caracterul transmis la invocarea constructorului, iar câmpul `leg` va deveni referința nulă `null` (către „nicăieri");

- metodele `parc_dir` și `parc_inv`.

Clasa `lista` conține metoda principală `main`, cu care începe executarea programului. Să presupunem că la intrare apare:

```
abc$
```

Este citit caracterul `'a'` și este creat un nou obiect de tip `elem`, al cărui câmp `c` este caracterul `'a'` și al cărui câmp `leg` este `null`. Variabila `u` de tip `elem` păstrează referința către acest obiect. Pentru claritate, să notăm cu P_1 acest obiect. Variabilele `p` și `u` memorează o referință la obiectul P_1.

În continuare este citit caracterul `'b'`. Apoi obiectul P_1 este „pus la lucru". El creează un nou obiect de tip `elem`, al cărui câmp `c` este caracterul `'b'` și al cărui câmp `leg` este `null`; fie P_2 acel obiect. Câmpul `leg` al lui P_1 este actualizat și devine P_2. Variabila `u` va conține acum o referință la obiectul P_2.

În continuare este citit caracterul `'c'`. Apoi obiectul P_2 este „pus la lucru". El creează un nou obiect de tip `elem`, al cărui câmp `c` este caracterul `'c'` și al cărui câmp `leg` este `null`; fie P_3 acel obiect. Câmpul `leg` al lui P_2 este actualizat și devine P_3. Variabila `u` va conține acum o referință la obiectul P_3.

Cum următorul caracter citit de la intrare este `'$'`, crearea listei de obiecte este încheiată.

Să analizăm efectul instrucțiunii

```
System.out.println(p.parc_dir(p));
```

care prevede scrierea la ieșire a șirului de caractere întors de invocarea `p.parc_dir(p)`.

Cum variabila `p` memorează o referință la obiectul P_1, obiectul P_1 va invoca metoda `parc_dir` cu parametrul P_1. Este întors șirul format din caracterul `'a'` la care se concatenează șirul întors de invocarea metodei `parc_dir` efectuată de obiectul P_2. Acest apel întoarce șirul format din caracterul `'b'` la care se concatenează șirul întors de invocarea metodei `parc_dir` efectuată de către obiectul P_3. Cum această din urmă invocare întoarce șirul vid (deoarece câmpul `leg` al lui P_3 conține `null`), rezultă că la ieșire va apare șirul `"abc"`.

❑

Conf. dr. Horia Georgescu este cadru didactic al Facultății de Matematică a Universității din București. E-mail: **hg@math.math.unibuc.ro**.

Glosar de termeni Java

instanță (*instance*) – Un obiect aparținând unei anumite clase. În programele Java, o instanță a unei clase este creată utilizând operatorul *new* urmat de numele clasei.

interfață (*interface*) – Un grup de metode care poate fi implementat de mai multe clase, indiferent unde se află aceste clase în ierarhia de clase.

interpretor (*interpreter*) – Un modul care decodifică și execută alternativ fiecare instrucțiune dintr-un anumit cod. Interpretorul Java decodifică și execută byte-code Java. Vezi și „compilator".

mașina virtuală Java (*Java virtual Machine*) – Componentă a mediului de executare Java (*Java runtime environment*) responsabilă cu interpretarea byte-code-ului Java.

metodă (*method*) – O funcție definită în interiorul unei clase. Dacă nu se specifică altfel, o metodă nu este statică (deci se invocă pentru o anumită instanță a clasei).

metodă statică (*class method* sau *static method*) – O metodă care este invocată fără referire la un anumit obiect. Metodele statice afectează clasa în întregime și nu o anumită instanță.

moștenire (*inheritance*) – Modalitatea prin care clasele conțin în mod automat câmpurile și metodele definite în clasele care se află deasupra lor în ierarhia de clase.

obiect (*object*) – principalul element constructiv al programelor orientate spre obiecte (*object-oriented*). Fiecare obiect este o unitate de program constând din date (instanțe ale câmpurilor) și funcționalitate (instanțe ale metodelor). Vezi și „clasă".

pachet (*package*) – un grup de tipuri. Pachetele sunt declarate cu cuvântul-cheie *package*.

proces (*process*) – un spațiu virtual de adrese conținând unul sau mai multe fire de executare.

sistem de rulare (*runtime system*) – Mediul software în care programele compilate pentru mașina virtuală Java pot fi executate. Sistemul de rulare cuprinde în întregime codul necesar pentru a încărca programe, pentru a lega dinamic metode native, pentru administrarea memoriei, tratarea excepțiilor, precum și o implementare a mașinii virtuale Java, care poate fi un interpretor Java.

subclasă (*subclass*) – O clasă care este derivată dintr-o altă clasă (deci situată cu unul sau mai multe niveluri mai jos în ierarhia de clase).

superclasă (*superclass*) – O clasă din care o anumită clasă este derivată (deci situată cu unul sau mai multe niveluri mai sus în ierarhia de clase).

(M.S.)

TIPURI de DATE

de Mihai Budiu

Un capitol esențial al programării, care este adesea abordat dintr-o perspectivă intuitivă, lipsită de rigoare. Acest articol vă propune o descriere precisă a tipurilor de date uzuale, folosind pentru aceasta limbajul matematicii.

Avantajele folosirii unui limbaj cu tipuri de date sunt apreciate de orice programator experimentat. Ele:

- ușurează înțelegerea programului de către om;
- încurajează o gândire și o formulare disciplinată a problemei;
- ușurează munca de optimizare a compilatorului;
- permit depistarea erorilor într-o fază timpurie.

Acest articol nu va trata aceste aspecte, ci își propune să explice într-un cadru cât se poate de riguros o colecție foarte simplă de fapte care descrie semnificația tipurilor de date. Descrierile uzuale ale acestui capitol – esențial – al informaticii păcătuiesc prin prezentări neriguroase, intuitive. Cea de față va tinde spre maximă precizie, folosind limbajul precis al matematicii.

Ce este un tip de date

Există mai multe definiții posibile pentru tipurile de date, din mai multe perspective. O privire globală definește simultan atât tipurile de date cât și operațiile care se pot face cu ele, pentru că acestea sunt de fapt cele două fațete ale unei aceleiași monezi: nu pot exista una fără cealaltă. Noi vom urca însă dealul pe partea mai lină și vom trata cele două subiecte pe rând și în mod inegal, acordând o oarecare prioritate tipurilor de date. Astfel, în cea mai simplă accepțiune posibilă, un tip de date nu este altceva decât: „*O mulțime de valori*".

Ce este valoarea? Nu are nici o importanță! Este un nume special dat elementelor care aparțin unui tip. Termenul „aparțin" este justificat pentru că, rețineți, tipul este o ***mulțime***, în cel mai precis sens matematic. Exemple? În Pascal tipul `boolean` este o mulțime cu două elemente. Aceste două elemente se notează în Pascal cu `false` și `true`. `False` și `true` se numesc de aceea „valori booleene". Tipul `integer` este tot o mulțime care include, printre altele, niște elemente (valori) care în Pascal se notează cu -2, -1, 0, 1, 2, 3, etc. Tipul `integer` este un tip interesant, pentru că își propune să mimeze mulțimea matematică Z a numerelor întregi, dar nu reușește prea bine, pentru că mulțimea „integer" este finită, iar Z nu! Care sunt limitele mulțimii „integer" nu se specifică. În dialectul Turbo Pascal (cel mai răspândit pe PC-uri) „integer" are 65536 de elemente, de la cel notat -32768 la cel notat 32767. Elementele tipului `integer` sunt numite valori întregi. Vom zice de aceea:

boolean = {false, true}

unde acoladele sunt notația uzuală pentru mulțime.

Tipuri enumerate

Pascal pune la dispoziție programatorului mai multe mecanisme prin care el să-și construiască noi tipuri. Cel mai simplu dintre acestea este ***enumerarea***. Enumerarea construiește un nou tip cu un număr finit de elemente descriind între paranteze reprezentarea canonică a fiecărui element. Exemplu (tipic):

(rosu, verde, albastru, galben, mov, negru, alb)

este descrierea unui nou tip care are 7 elemente cu reprezentările „rosu", „verde", etc. Putem da acestui tip un nume, folosind directiva `type` din Pascal.

```
type culoare =(rosu, verde,
 albastru, galben, mov, negru, alb);
```

Din punct de vedere matematic am putea scrie:

culoare = {rosu, verde, albastru, galben, mov, negru, alb}

Tipul `boolean` din Pascal ar putea fi definit el însuși astfel:

```
type boolean = (false, true);
```

De ce ne este oferit însă de-a gata vom afla din următoarea secțiune și din cele consacrate operatorilor.

Tipuri fundamentale

Limbajul Pascal conține patru tipuri gata construite. Pe acestea le numim „tipuri fundamentale". Ele sunt: `real`, `integer`, `char`, `boolean`.

E un exercițiu interesant să scriem mulțimile asociate fiecăruia dintre aceste tipuri. Din păcate nu este o treabă ușor de făcut, ci chiar imposibil pe alocuri. Pentru `boolean` am văzut:

boolean = {false, true}

Pentru `integer` am putea scrie ceva de genul:

integer = { n ∈ Z |
limita_de_jos ≤ n < limita_de_sus }

Limita_de_jos și *limita_de_sus* depind de dialectul de Pascal folosit. Pentru Turbo ele sunt (după cum am zis) -32768, respectiv 32768.

Pentru `char` lucrurile sunt grele pentru că există unele caractere care nu au o reprezentare evidentă! Tot ce putem spune este că această mulțime include mulțimea

{'a', 'b', 'c', ... 'A', 'B', ... , '0', ...,
'!', '@', '#', ...}

a caracterelor care pot fi tipărite.

Iar pentru `real` lucrurile sunt imposibile pentru că, deși `real` se dorește o imagine (izomorfă) a mulțimii R a numerelor reale, în realitate `real` este finită și are elementele distribuite într-un mod foarte ciudat pe axa reală (nici măcar două elemente ale ei nu sunt echidistante)! Studiul acestei mulțimi aparține analizei numerice, care studiază modul cum putem face calcule atunci când putem manipula numai aproximări ale valorilor care ne interesează.

O observație interesantă: deși în matematică Z este inclus în R, în Pascal `integer` nu este inclus în `real`!

Derivarea tipurilor

Pascal (și nu numai el) pune la dispoziție niște mecanisme puternice prin care putem construi noi tipuri plecând de la cele existente. (Prin enumerare putem construi tipuri plecând de la nimic.) Derivarea dă naștere unei noi mulțimi de elemente combinând elementele unor tipuri deja existente. Anumite operații nu se pot face cu orice tip.

Există o interesantă clasă de tipuri numite ***tipuri scalare***. Tipurile `boolean`, `char`, `integer` și toate tipurile construite prin enumerare sunt scalare. Aceste tipuri au întotdeauna un număr finit de elemente și după cum vom vedea mai târziu, există (datorită operatorilor de comparație) o relație de ordine totală pe ele (adică dintre două, unul e mai mare și celălalt mai mic).

Notă: Raportul Pascal marchează și tipul `real` ca fiind scalar, dar din motive de omogenitate noi nu îl vom considera astfel, pentru că el nu se comportă în nici o privință ca un tip scalar (nu poate fi indice de `array`, nu poate fi folosit într-o instrucțiune `case` ori `for` etc.)

Subdomeniu

Există o operație prin care dintr-un tip ***scalar*** putem extrage doar o submulțime formată din elemente consecutive. Ea devine un nou tip, care se va numi un ***subdomeniu*** (*subrange*) al celui inițial. Deoarece pe tipurile scalare există o ordine, submulțimea se indică prin capetele ei.

De exemplu:

```
1..100
```

este un tip subdomeniu al `integer`, având 100 de elemente. Iată un tip subdomeniu al tipului culoare definit mai sus, căruia îi dăm și un nume:

```
type CulPrimara = rosu..albastru;
```

Tipul numit `CulPrimara` are 3 elemente:

CulPrimara = {rosu, verde, albastru}

Tipurile subdomeniu sunt toate scalare.

Articol (record)

Dacă avem la dispoziție tipurile `T1`, `T2`, ... `Tn` putem forma un nou tip cu construcția

```
record
   c1 : T1;
   c2 : T2;
   ....
   cn : Tn
   end;
```

Ce mulțime reprezintă noul tip? O valoare de tip `record` subsumează n valori, fiecare din tipul `Ti` respectiv. Cu alte cuvinte, putem asimila un `record` cu un tuplu (o mulțime ordonată) cu n elemente (v1, v2, ..., vn), unde v1 ∈ T1, v2 ∈ T2, ..., vn ∈ Tn.

Să dăm un nume tipului de mai sus:

```
type R = record
   c1:T1;
   c2:T2;
   ...
   cn:Tn
   end;
```

atunci

R = { (v1, v2, ..., vn) |
v1 ∈ T1, ... vn ∈ Tn }

sau, altfel scris:

R = T1 x T2 x ... x Tn

unde x este produsul cartezian.

GLOSAR de termeni

TIP DE DATE (*DATA TYPE*)
- o mulțime de valori căreia adesea i se adaugă un set de operatori și un set de reguli de folosire a operatorilor

OPERATOR (*OPERATOR*)
- o operație între mai multe valori dând ca rezultat o altă valoare; strict vorbind o funcție (parțială) definită pe un produs cartezian de tipuri cu valori într-un tip

POLIMORFIC (*POLYMORPHIC*)
- un operator care operează la fel asupra mai multor tipuri de date structurate asemănător

SUPRAÎNCĂRCAT (*OVERLOADED*) [despre un operator]
- în realitate mai mulți operatori cu același nume, dar care sunt distinși după contextul în care se aplică (tipurile argumentelor și al rezultatului)

TIP SCALAR (*SCALAR TYPE*)
- un tip cu un număr finit de elemente asupra cărora este fixată o relație de ordine totală (cu ajutorul unor operatori)

Un tip construit cu record nu e scalar.

Notă: pentru deplina rigoare ar trebui să includem în mulțimea R și numele câmpurilor c1, c2, ..., cn. O definiție completă ar fi: produsul cartezian al tipurilor componente, având fiecare componentă etichetată cu numele corespunzător.

Vector (matrice)

Dacă avem un tip oarecare B și un tip scalar S (cu excepția lui integer) putem forma din ele un nou tip cu operația:

```
array [S] of B
```

Nu este prea complicat de văzut că asta înseamnă:

```
type A = array[S] of B;
```

$A = B \times B \times \ldots \quad B = B^n$

unde produsul este luat de cardinal din S (numărul de elemente al lui S) ori. Aceasta pentru că un array[S] este o colecție de atâtea valori câte are S de tipul B.

Un tip construit cu array nu e scalar.

Notă: Ca mai sus, fiecare componentă a unui element din produsul B la puterea n este etichetată cu o valoare corespunzătoare din S.

Mulțime

Dacă avem un tip scalar S putem forma tipul (dacă S nu are prea multe elemente):

```
set of S
```

Ce valori are acest nou tip? Un element al tipului set of S este o mulțime formată cu elemente din S. Deci dacă avem

```
type R = set of S;
```

atunci

$R = \{ M \mid M \subseteq S \} = P(S)$

adică mulțimea părților lui S.

Un tip construit cu set nu este scalar.

Fișier

Fișierul este un tip interesant. Este prima metodă de a construi un tip care – teoretic vorbind – este infinit! (Toate tipurile de până acum sunt finite pentru că sunt construite prin combinații finite ale unor mulțimi finite). Bineînțeles că practic și acest tip este finit, dar cu o dimensiune maximă neprecizată. Fiind dat un tip T (care nu e fișier la rândul său) ce face operația: file of T? Un fișier este asemănător unui array, dar are o lungime neprecizată. El conține 0 sau 1 sau 2, ..., sau n, ... valori de tipul T la un moment dat. Deci, dacă avem:

```
type F = file of T;
```

atunci, ca mulțime, avem:

F = reuniune după n { T la puterea n | n ∈ N }

unde N este mulțimea numerelor naturale. O astfel de operație aplicată lui T (reuniunea tuturor puterilor) se numește operația lui Kleene, și se notează cu un exponent steluță (T la puterea *).

Record cu case

Ajungem la o operație mai puțin omogenă de combinare a tipurilor. Să considerăm directiva Pascal:

```
type RC = record
    c1 : T1;
    case c2 : T2 of
      v1 : ( c3 : T3 );
      v2 : ( c4 : T4 )
    end;
```

Un model aproximativ este dat de

$RC \subseteq T1 \times T2 \times (T3 \cup T4)$

pentru că o valoare de tip RC are primele două câmpuri de tipurile T1, respectiv T2, și un alt câmp dintr-unul din tipurile T3 sau T4. În realitate RC nu este atât de vastă, pentru că nu orice combinație din mulțimea indicată este în RC. Când câmpul al doilea are valoarea v1, câmpul al treilea nu poate fi de tipul T4! Deci corect este:

$RC = T1 \times (\{v1\} \times T3 \cup \{v2\} \times T4)$

{v1} este mulțimea formată din elementul v1 aparținând lui T2. Un astfel de tip nu este scalar.

Pointer

Ultimul mod de derivare a tipurilor în Pascal este prin luarea referinței la ele. Un tip

```
type PT = ^T;
```

are printre valorile sale o valoare specială nil, și apoi valori care „arată" spre valori din tipul T. Tipul PT este deci izomorf (corespondent unu la unu) cu tipul T, la care se adaugă un nou element. Vom face mai mult, le vom considera egale:

$PT = T \cup \{ nil \}$

Tipul pointer din Pascal este singurul tip care poate să includă în definiția sa pe sine însuși, ca în:

```
type PA = ^A;
     A = record
           i : I;
           n : PA
         end;
```

Asta dă o interesantă „ecuație" pentru mulțimea PA:

$PA = \{nil\} \cup A = \{nil\} \cup (I \times PA)$

Nu este nimic contradictoriu în această formulă! Cele două mulțimi sunt într-adevăr egale, în sensul că fiecare element care se află în PA este fie nil fie se poate scrie sub forma *(i, pa)* unde $i \in I$ și $pa \in PA$ și reciproc: orice valoare de forma (*i*, *pa*) este o listă. De fapt aceasta este definiția (recursivă) a unei liste (ați recunoscut probabil tipul PA ca fiind tipul celulei de bază care compune o listă simplu înlănțuită):

O listă (de tipul I) este fie lista vidă, fie o valoare de tipul I împreună cu o listă „mai mică" de același tip.

Numele tipurilor

Am folosit adesea în exemplele anterioare nume asociate tipurilor cu ajutorul directivei type. În construcția:

```
type t = record
     a : array[1..10] of integer;
     c : char
     end;
```

avem două părți: 1) una care construiește un nou tip:

```
record
     a: array[1..10] of Integer;
     c: char
     end;
```

2) una care îi dă un nume acestei construcții:

type t = *construcție*;

Acestea sunt două operații distincte. Mai mult, în Pascal tipurile t și record a:array[1..10] of integer; c:char end sunt în realitate distincte, deși au aceeași structură (sunt formate prin produsele acelorași mulțimi):

integer la puterea 10 x char

Asta putem vedea dacă scriem:

```
var x, z: t;
        y: record
          array[1..10] of integer;
          c: char
        end;
```

Acum x și y au tipuri diferite! O instrucțiune de felul x:=y nu este permisă! Putem face însă x:=z! Deci: două tipuri nu sunt „compatibile" chiar dacă au aceeași structură. Ele trebuie să aibă același *nume*, și atunci de fapt reprezintă același tip.

Operatori

În realitate orice definiție a unui tip ca mulțime este sortită eșecului, pentru că un tip include și o mulțime de operații pe valorile sale care iau naștere automat odată cu crearea sa. De exemplu, de îndată ce creăm un tip enumerat avem la dispoziție o operație de comparație între elementele sale.

Ce este o operație? Aparent banală, întrebarea nu are un răspuns evident. Div este o operație între doi întregi. Ce înseamnă asta? Ea combină cele două valori întregi cărora li se aplică pentru a genera una nouă. Este deci o *funcție* definită astfel:

div : integer x integer → integer

Ea asociază perechii (a, b) de numere întregi numărul „câtul împărțirii lui a la b". Pe deasupra, este o funcție care nu se poate calcula pentru toate perechile de întregi. Dacă al doilea este 0 atunci operația nu are sens. Deci funcția este *parțială*, adică nu este definită pentru toate valorile posibile ale argumentelor.

De fapt așa sunt toți operatorii – funcții din tipuri (sau produse carteziene de tipuri) în tipuri. De ce sunt tipurile atât de legate de operatori? Pentru că crearea unui nou tip prin oricare din operațiile de mai sus dă naștere în mod automat unor operatori care lucrează cu valori din acel tip. De exemplu definirea unui tip enumerare cum este „culoare" de mai sus dă naștere următorilor operatori:

<, >, <=, >=, <>, = : culoare x culoare → boolean
succ, pred : culoare → culoare
ord : culoare → integer

Unii operatori nu par a fi chiar operatori datorită sintaxei deosebite a folosirii lor; unii se scriu între argumentele lor (ca <), alții înainte (ca ord),

Operatori

Operatori pentru BOOLEAN

=, <>, <, >, <=, >= : boolean x boolean → boolean
and, or : boolean x boolean → boolean
not : boolean → boolean
succ, pred : boolean → boolean
ord : boolean → integer

Operatori pentru INTEGER

=, <>, <, >, <=, >=: integer x integer → boolean
succ, pred : integer → integer
ord : integer → integer
+, -, *, div, mod : integer x integer → integer
odd : integer → boolean
sqr, abs : integer → integer

Operatori pentru CHAR

=, <>, <, >, <=, >= : char x char → boolean
succ, pred : char → char
ord : char → integer
chr : integer → char

Operatori pentru REAL

+, -, /, * : real x real → real
=, <>, <, >, <=, >= : real → boolean
sin, cos, ln, exp, abs, sqrt, sqr, arctan : real → real

Pentru orice tip ENUMERARE (T)

pred, succ : T → T
=, <>, <, >, <=, >=: T x T → boolean
ord : T → integer

Pentru orice tip SUBRANGE (T)

moștenește toate operațiile tipului din care este derivat

Pentru orice tip ARRAY (T = array[S] of B)

[.] : T x S → B (operația de indexare aplicată astfel: t[s], unde t aparține lui T și s aparține lui S)

Pentru orice tip MULȚIME (T = set of S)

=, <>, <=, >= : T x T → boolean
in : S x T → boolean
+, -, * : T x T → T
[] : S la puterea * → T (constructorul de mulțime aplicat astfel: [s, s_1, s_2,..., s_3] unde s_i aparțin lui S - această exprimare este discutabilă).

Pentru orice tip RECORD (R = record e1:T1; ... en:Tn end)

. : R x mulțimea etichetelor → reuniune după i din Ti (operația de selecție, care se aplică astfel: r.e, unde *r* aparține lui R și *e* este una din etichetele ei)

Pentru orice tip POINTER (PA = ^A)

^ : PA → A (operația de dereferențiere, aplicată astfel: pa^, unde *pa* este de tipul PA)

Pentru tipul FIȘIER (F = file of I)

eof : F → boolean

alții în feluri mai ciudate (ca [ ] pentru indexat array-uri, care se scrie vis-a-vis de cele două argumente ale sale a și n astfel: a[n]).

În caseta „Operatori" puteți vedea pentru fiecare tip ce operatori are (dacă este fundamental) și căror operatori le dă naștere.

În plus, raportul Pascal specifică faptul că accesul la un fișier se face (la nivelul cel mai de jos) prin două proceduri put și get, care manipulează o valoare de tip I asociată fișierului, numită buffer. Astfel, pentru a scrie în fișierul f:file of I; valoarea variabilei a de tip I facem:

```
f^ := a;
put(f);
```

iar pentru a citi în variabila a de tip I

```
a := f^;
get(f);
```

Prima secvență este echivalentă cu instrucțiunea write(f, a), iar a doua cu read(f, a);

În principiu, nu există operatori pentru lucrul cu fișiere. Putem aproxima cu un operator operația prin care accesăm valoarea bufferului asociat unui fișier, notată mai sus ca la pointer, cu ^, după cum urmează:

^ : F → I

Polimorfism și supraîncărcare

După cum vedeți, anumiți operatori sunt definiți pentru mai multe tipuri. De exemplu, succ merge pe orice tip scalar, dând un rezultat de același tip. În realitate

succ : integer → integer

și

succ : char → char

sunt doi operatori diferiți dar care au același nume. Un astfel de operator, care funcționează pe mai multe tipuri de date simultan și al cărui mod de comportare depinde de tipul la care se aplică, se numește „supraîncărcat". Adesea sunt bineveniți, adesea pot să dea naștere la probleme. De exemplu, teoretic există patru operatori „+", astfel:

+ : integer x integer → integer
+ : real x real → real
+ : integer x real → real
+ : real x integer → real

Toți aceștia sunt supraîncărcări ale lui „+". (Ca să nu vorbim de folosirea lui „+" ca reuniune de mulțimi).

Putem să aplicăm unul și să ne gândim la altul, cum ar fi:

```
var
    i,j : integer;
    r : real;
    ...
    i := j + r;
```

Ori asta nu se poate, pentru că aici „+" este presupus a fi

+ : integer x real → integer

care nu există!

Există un gen special de operatori care seamănă cu cei supraîncărcați dar care nu sunt supraîncărcați ci *polimorfici*. Distincția între ei nu este câteodată foarte precisă. În Pascal, la acest titlu aspiră de pildă operatorii „." (punct, pentru selecție), [ ] (pentru indexare) etc. De ce? Pentru că operatorul [ ] este definit pentru orice tip. Dacă

R = **array** [S] **of** B

atunci

[ ] : R x S → B

și operează la fel, independent de natura tipurilor S (scalar) și B, extrăgând din valoarea de tipul R (un array) pe cea etichetată cu valoarea dată de tipul S.

Putem spune că [ ] este un singur operator, oricare ar fi tipul R format ca mai sus. [ ] este atunci un operator polimorfic – pentru că lucrează cu mai multe tipuri, dar în exact același mod.

Ca lucrurile să nu fie atât de simple, să observăm că mai există un operator notat tot cu [ ] pentru construit mulțimi, care este tot polimorfic! Cei doi operatori sunt distinși după modul în care se scriu în expresie. Acesta este supraîncărcare!

Concluzie

Lucrurile fundamentale din acest articol sunt puține. Totul poate să pară banal, însă de fapt nu este așa – o astfel de abordare riguroasă permite apoi efectuarea de raționamente matematice precise asupra programelor, pentru că obiectele cu care programele operează sunt definite precis.

Deci:

- un tip este o mulțime de valori. O variabilă are un anumit tip dacă valoarea ei în orice moment este un element al acelei mulțimi.
- un operator este o funcție care are argumente una sau mai multe valori și al cărei rezultat este tot o valoare.
- există operatori care lucrează în exact același mod pe mai multe tipuri. Aceștia se numesc polimorfici.
- există operatori diferiți dar care se notează la fel. Aceștia se numesc supraîncărcați.

❑

Mihai Budiu este doctorand la Carnegie Mellon University, SUA. E-mail: mihaib@cs.cmu.edu. *Web:* http://www.cs.cmu.edu/mihaib.

Indicații bibliografice

- Limbajul Pascal (standard) este descris în „**Pascal – user manual and report**", Kathleen Jensen, Niklaus Wirth, tipărit la Springer Verlag în 1974. Nu e o carte prea ușor de citit.

- Lucrurile de mai sus sunt excelent expuse într-o carte de bază pentru orice informatician. Ea este scrisă de Niklaus Wirth și conține o sumedenie de fapte extrem de interesante și bine explicate. Nu știu să existe vreo traducere în română. Se numește „**Algorithms + Data Structures = Programs**". Este publicată în 1976 la editura Prentice-Hall.

- O altă abordare interesantă, care integrează și operatorii în definiții axiomatice ale tipurilor este în cartea lui E. Horowitz și S. Sahni „**Algorithms and Data Structures**" publicată în 1978.

phone/fax: +40-62-436705, 436729
fax: +40-62-415574, 415156
telex: 033327 tenet r
e-mail: mark@multinet.eunet.ro

multinet

MULTINET Ltd.
11A/J UNIRII Blvd.
4800, Baia-Mare

DISTRIBUITOR DIRECT AL PRODUSELOR SYRACUSE LANGUAGE SYSTEMS

Your WayTM 2.0

Pentru afaceri, pentru călătorii sau doar pentru divertisment, *Your Way 2.0* este o cale extraordinară de a învăța o limbă străină. Participați la conversații realistice sau perfecționați-vă pronunția cu modulele de recunoaștere a vorbirii și înregistrare/redare; învățați gramatică, conjugarea verbelor și peste 2600 cuvinte cu ajutorul jocurilor angajante și educative.

Pentru vorbitorii unei alte limbi decît engleza există *Your Way 2.0 English (Immersion)*, care prezintă un format multimedia angajant și instrucțiuni simple pentru învățarea ușoară cuvinte cu ajutorul jocurilor multimedia cu grafică deosebită și pot conversa interactiv cu personaje de pe ecran. Recunoașterea vorbirii recompensează pronunția corectă.

Modulul de înregistrare/redare îi compară pe cei ce învață cu vorbitori nativi. *TriplePlay Plus!* folosește o pătrundere unică, interactivă, în

în care trebuie să sune vorbirea unei limbi străine. Trei nivele de joc asigură distracții și provocări atractive întregii familii.

Let's Talk

O cale ușoară de învățare a vocabularului, de îmbunătățire a pronunției și de realizare a fluenței în vorbirea unei limbi străine. *Let's Talk* ne învață peste 2200 cuvinte în spaniolă, franceză, germană sau italiană. Cei ce învață ascultă vorbitori nativi și urmăresc ilustrații ale cuvintelor și frazelor pe măsură ce utilizează jocuri interactive și folosesc glosarul multimedia. Fiecare secțiune a *Let's Talk* include îmbunătățirea accentului,

All-in-One Language Fun!

O introducere excitantă în 5 limbi străine (spaniolă, franceză, germană, japoneză, engleză) pe un singur CD-ROM!

Copiii se pot distra învățînd peste 200 cuvinte și fraze în fiecare limbă cu ajutorul versiunilor multimedia ale celor 27 de jocuri familiare: Jigsaw Puzzles, Bingo, Memory Teasers, Simon Says și multe altele. Jocurile educative prezintă grafică color, înregistrări ale vorbitorilor nativi și multe nivele de îndemînare. Ideal pentru cei mici, nu se cer aptitudini de a citi sau de a lucra cu tastatura.

Învățați *LIMBI STRĂINE* cu ajutorul calculatorului !

și plăcută a gramaticii, vocabularului, vorbirii, ascultării și conversației.

TriplePlay Plus!

Cel mai premiat program de învățat limbi străine publicat în toate timpurile, *TriplePlay Plus!* îi lasă pe cei ce învață: să asculte, citească și să vorbească într-o limbă nouă imediat. Utilizatorii învață sute de limbajul multimedia care vă va permite să învățați exclusiv în noua limbă, va oferi nenumărate oportunități de a învăța și repeta ceea ce ați învățat si ce este mai important și mai greu de realizat în lipsa unui partener, vă va ajuta să deprindeți vorbirea corectă într-un mod distractiv. Vocea vorbitorilor nativi este clară și naturală, și prezintă exemple despre modul modulele integrate de recunoaștere a vorbirii și de înregistrare/redare ajutînd la îmbunătățirea ușoară și efectivă a vorbirii.

Let's Talk English îi învață peste 2500 cuvinte și fraze pe vorbitorii de limbă chineză, franceză, germană, italiană, coreană, portugheză și spaniolă. *Let's Talk Inglés* oferă aceleași facilități vorbitorilor de limbă spaniolă.

Vocabulary	Listening Comprehension	Speaking	Speech Recognition	Reading Comprehension	Conversation	Verb Conjugation	Grammar and Usage
ALL-IN ONE For Kids							
LET'S TALK™ Skill Builder							
TRIPLEPLAY PLUS! Basic Course							
YOUR WAY™ Complete Course							

KASPAROV învins de DEEP BLUE...

de Ionuț Mușlea

Calculatoarele care joacă șah nu mai sunt de mult o noutate. Nici chiar victoriile acestora în fața oamenilor. Dar când însuși campionul mondial este învins de către un calculator, putem spune că ne aflăm în fața unui eveniment important. Doar înțelegând puterea acestui calculator vom înțelege imensa putere a minții umane.

7 iulie 1997, orele 16:30. Sala mare a Centrului de Conferințe din Providence, Rhode Island este arhiplină. Norocoșii au mai apucat un scaun, dar și cei aflați în picioare se consideră „răsfățați ai soartei": peste câteva secunde începe una dintre cele mai impresionante ceremonii din lumea inteligenței artificiale! La invitația organizatorilor, rând pe rând urcă pe scenă Richard Greenblatt, David Slate și Lawrence Atkin, Ken Thompson și Joe Condon, Hans Berliner, Carl Ebeling, Gordon Goetsch și Murray Campbell. Fiecare dintre ei primește „Medalia Allen Newell" pentru contribuții fundamentale la dezvoltarea șahului pe calculator. În fine, sunt invitați pe scenă eroii zilei, „părinții" lui Deep Blue: Feng H. Hsu, Murray Campbell, Thomas Anantharaman, Andreas Nowatzyk și Mike Browne. La rându-le, primesc și ei „Medalia Allan Newell", împreună cu „Premiul Fredkin" (în valoare de 100.000 dolari!) pentru victoria lui Deep Blue asupra lui Kasparov. Toată sala e în picioare, se aplaudă furtunos minute în șir, ba chiar unii își șterg pe furiș câte o lacrimă.

În mai, imediat după victoria lui Deep Blue, pe diversele „newsgroups" și „mailing lists" legate de IA (Inteligență Artificială) au început să curgă mesaje entuziaste legate de marea victorie. Era un entuziasm molipsitor, în care au fost angrenați și unii dintre „monștrii sacri" ai IA-ului, care se felicitau între ei pentru contribuții, perseverență și „credința în victoria finală". Cuvântul de ordine era: „nu mai există dubii: avem un *program inteligent* care l-a învins pe Kasparov, cel mai mare jucător de șah al tuturor timpurilor". Să fie oare adevărat?

Deep Blue, acest Kasparov al calculatoarelor

Șahul pe calculator a luat naștere odată cu articolul „Programarea unui calculator pentru a juca șah" (Claude Shannon, 1950). Domeniul a luat un avânt extraordinar și multe celebrități din lumea informaticii au contribuit la ceea ce urma în final să devină victoria lui Deep Blue. Pentru a cita doar câteva nume, îi amintim aici pe Alan Turing (1951, proiectarea primului program de jucat șah), Alan Kotok și John McCarthy (1966, meciul pierdut cu 3 la 1 în fața sovieticilor de la ITEP), Richard Greenblatt (*MacHack*, primul program are succes într-un „turneu uman", 1967), David Slate, Larry Atkin și Keith Gorlen (în 1967 *CHESS* câștigă primul loc în campionatul statului Minnesota), Ken Thompson și Joe Condon (programul *Belle* atinge în 1983 nivelul de maestru).

În 1986, Feng H. Hsu și Thomas Anantharaman creează CHIPTEST-ul („strămoșul" lui Deep Blue), un procesor specializat pentru șah care era capabil să genereze 10.000 de poziții pe secundă („calculatorul-

HotSoft

KeyMap &Fonts

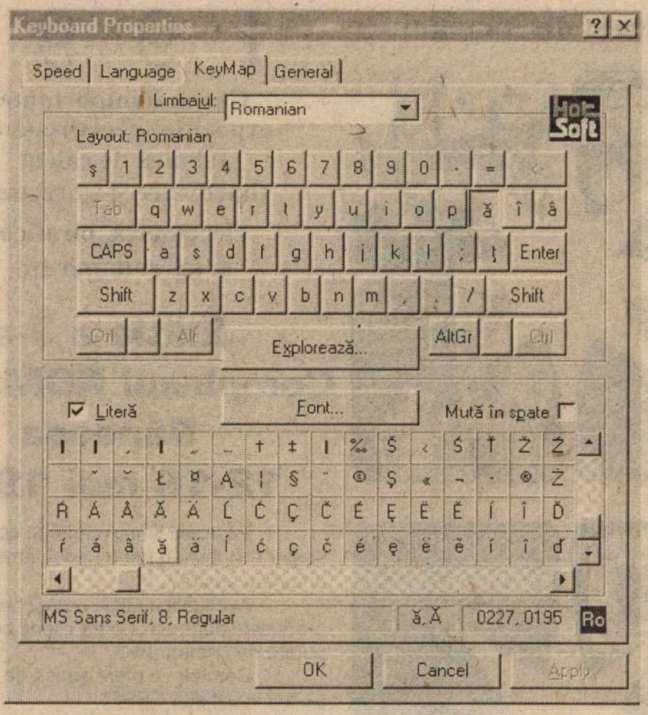
Add on

Designed for MICROSOFT® WINDOWS'95

VERSIUNEA 2

KeyMap 2.0

- este utilitarul Windows 95 de care aveți nevoie pentru a scrie comod în orice limbă.
- permite configurarea software a tastaturii în funcție de necesitățile specifice oricărui utilizator.
- pentru fiecare layout atașat unei limbi și pentru fiecare tastă a acestuia, puteți configura 4 caractere ce să fie generate de acesta!
- se instalează și se configurează foarte ușor pentru oricine a mai utilizat un produs Windows 95.
- Vă oferim de asemenea, la cerere, peste 180 de fonturi True Type sau Adobe Type (postscript), conținând toate caracterele specifice limbilor europene cu alfabet latin.



40(0)65-166516

***

www.hotsoft.ro/keymap

HotSoft SRL

Noul KeyMap 2.0 pentru Win95

împreună cu peste 180 de fonturi

Doresc să cumpăr produsul, KeyMap2.0.

V-am expediat la BCR Tg.Mureș în contul nr. 4072032198 al S.C.HotSoft SRL Tg-Mureș, 100.000 lei (în care sunt incluse și taxele poștale), cu mandatul poștal/ordinul de plată nr.: ..... din data de ....., pentru o licență a acestuia.

(Pentru licențe multiple apelați la sediul firmei.) ☐ Doresc să-mi trimiteți și factură.

Numele și prenumele: ..... Funcția: .....

Firma: .....Adresa: .....

Cod poștal/Localitate: ..... Telefon/Fax: .....

Data ..... Semnătura .....

COMTEK
EXPOSITIONS

vă invită la:

Expoziția și Conferința Internațională de Tehnologia Informației și Comunicații din România

Computers and Electronics Romanian Fair

CERF

1998

Cea mai importantă expoziție specializată de tehnică de calcul (hardware și software), electronică, birotică și telecomunicații

Ediția a 7-a
Complexul ROMAERO
Băneasa
12-16 mai 1998

Vă irosiți timpul căutând de unul singur distribuitori pentru echipamentele dumneavoastră?

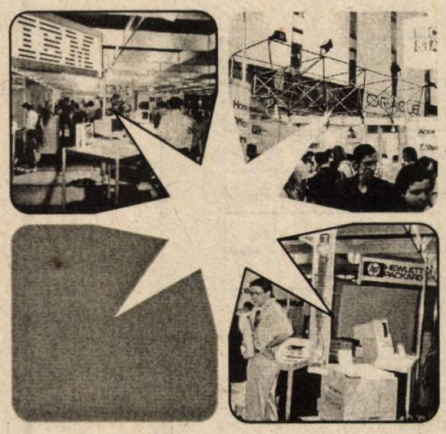
Aveți dificultăți în a găsi pentru produsele dumneavoastră clienți cu mare putere de cumpărare?

Comtek vă oferă șansa de a-i întâlni pe toți aceştia la CERF!

Organizată la puțin timp după CeBIT Hanovra, cea mai mare expoziție europeană în acest domeniu, CERF vă prezintă echipamente și tehnologii de ultimă oră.

Nu pierdeți ocazia de a veni la CERF!

Pentru informații sunați acum la
tel.: 231.18.82 / fax: 230.12.53;
tel./fax: 230.62.81



COMTEK
EXPOSITIONS

vă invită la:

Computers and Electronics Romanian Fair

Expoziția și Conferința Internațională de Tehnologia Informației și Comunicații din România

Ediția a 7-a

Complexul ROMAERO Băneasa
12 - 16 mai 1998

CERF
1998

CU PARTICIPAREA:

3M
A&C International S.A.
MobiFon
CLEPSYDRA
AAGNOR
AmTel
ALADDIN The Professional's Choice
ARIS ELECTRONIC
ASTI CONTROL
Autodesk.
Allied Telesyn International
ALLTROM
AGCR The Computer Company
AGIS COMPUTER
BASF
Bay Networks
BENTLEY The People Behind MicroStation
COMPAQ
Canon
CAPTOR
CIEL!
CHIP
VOGEL PUBLISHING

AGORA
compaCte COMPUTERS
CRESCENDO
Sun microsystems
Network Computing
Informatica Secolului XXI
ROMSYS SA
ELECTRONIC DESIGN & CONSULTING GROUP
FLAMINGO COMPUTERS
FORTE COMPUTERS Flexibility Made in Romania
Best Power
GEO STRATEGIES
HEWLETT PACKARD
IDG INTERNATIONAL DATA GROUP
IBM
ALO GSM
LYNX
FTP SOFTWARE
MOBIL ROM
Olympus

INFORMIX
K TECH
AT&T
MINOLTA
LASTING
ETA - 2U
LONGSHINE
mita
Metrologic
GENSYS Software Romania
Profuse Romania
MGT Trade srl
Microsoft WHERE DO YOU WANT TO GO TODAY?
IMATION
RADCOM
dialog
CYBERNETICS
Computervision
scala România
YAMAHA

ORACLE
UNISOFT ROMANIA
Pitney Bowes
PHILIPS
InTech
NCR
Q'NET
VLAMIR
REXEL
ROMSYM DATA SRL
IMA Infoconsult
Tape Computer
ROMUS INDUSTRIES
ILS Holding
unitech
FAST 2000
Sirom Graphics Computers S.A.
MAGIC Systems
TOTALSOFT
SOWAH
Hitachi

SPRINT
CONNEX
star MICRONICS
SCO
PRINTRONIX
SYSTEM PLUS
TEC
Thunder BYTE
TOTAL Technologies
Sistems
TREND
TRIPP LITE THE POWER PEOPLE
VIEWSONIC
SYMANTEC
EDEN SOFT
US Robotics
Verbatim Data Life Plus
VERO VERO ELECTRONICS
ComputerLand ROMÂNIA
WIZROM
WYSE

Pentru informații sunați acum la tel./ fax: 230.62.81; fax: 230.12.53; tel.: 231.18.82

COMPUTER PRESS
AGORA

Prin intermediul firmei noastre puteți primi în mod regulat oricare dintre revistele americane de calculatoare din tabelul alăturat. Pentru aceasta este suficient să ne trimiteți o comandă în care să specificați titlurile dorite și datele Dvs. de identificare (nume, adresă, telefon, fax), iar noi vă vom expedia cu plata prin ramburs revistele solicitate. Prețul în lei pe care îl veți achita la primirea fiecărui colet poate fi obținut prin înmulțirea prețului în dolari specificat în tabel cu cursul de schimb valutar al BNR la care se adaugă cheltuiala de expediție percepută de Poșta Română. Atenție deci,

NU TRIMITEȚI BANI ÎN AVANS!!!

Vă rugăm să expediați comenzile Dvs. pe adresa:

Computer Press Agora srl, CP 94 OP 49, București, fax: 3309285

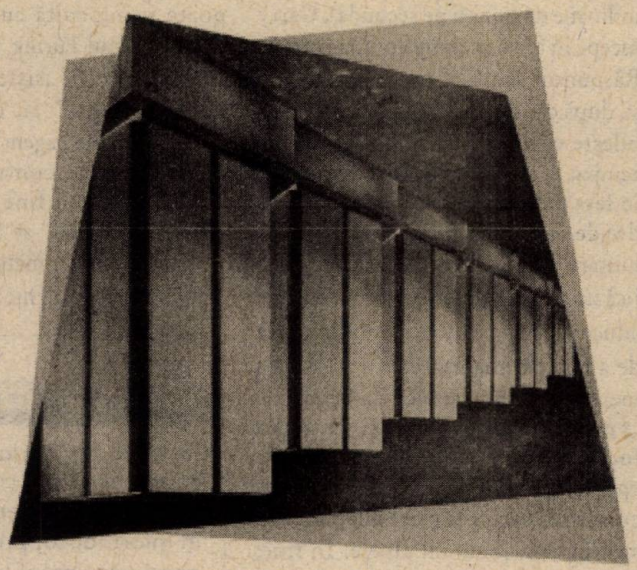
TITLU	FRECVENȚĂ	PREȚ
3D DESIGN	12	5.93
ACCESS OFICE VBA ADVISOR	12	7.49
AIXTRA	4	7.43
BOARDWATCH	12	8.93
BYTE	12	5.93
C/C++ USERS JOURNAL	12	7.43
C++ REPORT	9	11.93
UNICENTER TNC ADVISOR	6	8.93
CADALYST	12	5.93
CADENCE	12	5.93
CD-ROM PROFESSIONAL	12	10.43
CIRCUIT CELLAR	12	5.93
COMPUTER ARTIST	6	7.43
COMPUTER GAMING WORLD	12	5.99
COMPUTER GAMING WORLD w/CD	12	11.99
COMPUTER GRAPHICS WORLD	12	7.43
COMPUTER PLAYER	12	10.43
COMPUTER SHOPPER	12	7.49
COMPUTER SHOPPER BUYERS CD		7.49
CYBER SPORTS	4	7.49
DATA BASE WEB ADVISOR	12	7.49
DATABASE PROG. & DESIGN	12	7.43
DBMS	12	5.93
DBMS 1996 BUYERS GUIDE		7.43
DELPHI INFORMANT	12	8.93
DIGITAL VIDEO MAGAZINE	12	7.43
DR. DOBBS JOURNAL	12	5.93
ELECTRONIC PUBLISHING	12	7.43
ELECTRONIC GAMING MONTH	12	7.49
EMG2	12	7.49
FILE MAKER ADVISOR	24	7.99
FOXPRO ADVISOR	12	7.49
GAMEFAN	12	8.93
GAME DEVELOPER	6	8.93
ULTRA GAME PLAYER	12	7.49
GAME PRO	12	7.49
INTERACTIVITY	12	7.43
INTERNET WORLD	12	7.43
INTERNET JAVA & ACTIVEX ADVISOR	6	10.49

TITLU	FRECVENȚĂ	PREȚ
JAVA DEVELOPERS JOURNAL	12	11.93
JAVA PRO	4	10.43
JAVA REPORT	6	7.43
LAN MAGAZINE	12	7.43
LINUX JOURNAL	12	6.00
LOTUS NOTES ADVISOR	12	10.49
MACADDICT	12	11.99
MACTECH	12	8.98
MACWORLD	12	8.98
MAGAFAN	6	5.93
MICROSOFT INTERACTIVE DEV	12	7.43
MICROSOFT SYSTEMS JOUR.	12	7.43
MONDO 2000	4	8.93
NEW MEDIA	12	7.43
OBJECT MAGAZINE	6	6.75
OBJECT ORIENTED PROGRAM.	9	13.50
PC COMPUTING	12	5.99
PC GAMER	12	5.99
PC GRAPHICS & VIDEO	12	7.43
PC MAGAZINE	22	5.99
PC MAGAZINE CD-ROM	4	26.93
PC MAGAZINE SPECIAL ISSUE	4	8.93
PC WORLD	12	8.93
PRACTICAL WINDOWS	6	8.93
PUBLISH	12	7.43
SOFTWARE DEVELOPMENT	12	5.93
SYS ADMIN	6	7.43
UNIX REVIEW	12	5.93
VB TECH JOURNAL	12	5.25
VIDEO TOASTER USER	6	7.43
VISUAL BASIC PROG. JOUR.	12	5.93
VISUAL DEVELOPER MAGAZINE	6	7.43
WEB, THE	12	5.93
WEB DEVELOPER	6	8.93
WEB TECHNIQUES	12	7.43
WINDOWS/DEV DEVELOPERS	12	7.43
WINDOWS NT	12	7.43
WINDOWS SOURCES	12	4.49
WINDOWS TECH JOURNAL	12	6.00
WIRED	12	7.43

Revistele pot fi răsfoite, comandate și cumpărate de la librăria noastră deschisă la sediul firmei ***Forte Computers****: str. Lipscani nr. 102; precum și de la următoarele magazine din* ***București****: Flamingo Computers, Bd.N. Titulescu nr.102; MIVAS: Pasaj Universitate; Raffles Computers Shop, Calea Victoriei nr.25; Librăria "Noi", sala Dales; Librăria Humanitas din Pasajul Cretzulescu (lîngă magazinul Muzica);* ***BRAȘOV****: Interprof International, str.Operetei nr.86;* ***TIMIȘOARA****: XCOMP21 SRL Bul Regele Ferdinand nr. 2*

Puteți comanda orice carte de la: ***Computer Press Agora s.r.l.****, C.P. 94 OP.49, București, fax: 01-3309285*

Important: Lista completă de prețuri poate fi obținută via BBS 065-210780 (login: bbs, terminal: ANSI, ...)



Față în față: Garry Kasparov - considerat de mulți drept cel mai mare șahist al tuturor timpurilor - și IBM Deep Blue, cea mai puternică mașină de jucat șah construită. Până acum.

gazdă" fiind o stație Sun 3). După un an, CHIPTEST urca la 500.000 poziții pe secundă, iar în 1988 noul lor sistem (**Deep Thought**) genera peste 700.000 de poziții pe secundă, avea o „carte de deschideri" de 5000 de poziții și rula pe Sun 4. În 1989, Deep Thought lucrează pe o mașină cu șase procesoare, generează două milioane de poziții pe secundă și începe să joace „sub culorile" IBM. În același an are loc primul meci Deep Thought - Kasparov, dar „hardughia" este (încă) o pradă ușoară pentru Garry. Următorul pas l-a reprezentat Deep Thought II, care în 1991 rula pe un IBM RS/600, utiliza 24 de procesoare specializate și putea genera aproape 7 milioane de poziții pe secundă.

Anul 1994 aduce câteva noutăți de marcă în lumea șahului pe calculator: în februarie se naște „Deep Blue Prototype", în mai Kasparov pierde (din cauza lipsei de timp) o partidă cu programul FRITZ 3, iar în august Kasparov mai pierde o partidă și în fața programului *Pentium Chess Geniues* (Richard Lang). Odată cu primele evoluții în lumea reală, încep și problemele lui Deep Blue (DB): în iunie pierde un meci din cauza unei... furtuni ce a influențat funcționarea sistemului, iar în mai 1995, în timpul analizei unei situații critice (un *bug* „l-a scos" pe DB prea devreme din cartea de deschideri), linia telefonică se întrerupe (în America anului 1995!) și DB pierde din cauză că nu mai apucă să refacă căutarea începută înainte de „căderea" liniei telefonice.

În fine, în februarie 1996 începe marele meci cu Kasparov. Din cauza problemelor tehnice, Deep Blue va juca cu doar 192 de procesoare specializate (față de cele 1024 proiectate). Calculatorul-gazdă e un IBM RS/6000 cu 32 de procesoare, iar Deep Blue generează nu mai puțin de o sută de milioane de poziții pe secundă. Dar performanțele tehnice pălesc în fața „războiului nervilor" declanșat de către echipa IBM: americanii îi refuză lui Gary câteva jocuri de încălzire și păstrează secret numele persoanei ce se ocupă de „deschiderile" lui Deep Blue (în 1995, IBM îl angajase pe marele maestru internațional Joel Benjamin atât pentru jocuri-test, cât și pentru îmbunătățirea deschiderilor). În plus, IBM-ul găsește soluția perfectă pentru ca Garry să nu aibă nici cea mai vagă idee despre oponent: deoarece cele două părți semnaseră în mai 1995 un protocol prin care se angajau să schimbe informațiile legate de **toate** partidele oficiale, Deep Blue nu a mai jucat *nici o partidă oficială* după acea dată!

După ce – spre surpriza generală – Garry pierde prima partidă, în cea de a doua întâlnire încep problemele IBM-iștilor: mai întâi Deep Blue nu se inițializează corect și, după a doua mutare, „o ia razna", apoi la mutarea a noua se face o eroare de tastare, iar ceva mai târziu se pierde legătura telefonică! Garry câștigă cea de a doua partidă, după care urmează două remize. În cea de a cincia partidă are loc un eveniment semnificativ pentru modul în care Deep Blue joacă șah: Garry propune o remiză, calculatorul se „auto-analizează" și vede că e într-un ușor dezavantaj, dar decizia finală le aparține... oamenilor de la IBM, printre care și marele maestru internațional Joel Benjamin. Aceștia decid să ***nu*** accepte remiza, așa că partida continuă și Kasparov câștigă! În ultima partidă, Garry avea nevoie doar de o remiză, dar Deep Blue „crapă" la mișcarea 20 (și pierde astfel minute prețioase cu reinițializarea) iar Garry câștigă atât partida cât și meciul (scor final: 4-2).

La mai puțin de 18 luni din momentul în care Kasparov „salva onoarea omenirii" în fața automatului de jucat șah, mai precis la 3 mai 1997, începe super-revanșa. Deși Deep Blue

a mai „pus mușchi pe el" (cântărește 1,4 tone și calculează între 200 și 300 milioane de poziții pe secundă), Garry începe în forță și câștigă prima partidă. Răspunsul „bestiei" nu se lasă așteptat și, după cea de a doua partidă, se restabilește egalitatea. Urmează apoi trei remize, și iată-i pe cei doi competitori în fața partidei numărul șase, care va decide soarta meciului. După a șaptea mutare a lui Garry, Deep Blue sacrifică un nebun pentru un pion, fapt ce-l aduce pe Kasparov în imposibilitatea de a mai efectua rocada. În acest moment, conform spuselor lui Seirawan, „fața lui Kasparov exprima un singur lucru: **teroare**". Garry este atât de epuizat încât cade într-o capcană arhicunoscută care face parte din propria-i strategie în jocul cu albele. În fine, Kasparov se recunoaște învins la mutarea 19 și scorul devine 3.5 - 2.5 în favoarea lui Deep Blue, care câștigă astfel meciul!

Este Deep Blue inteligent?

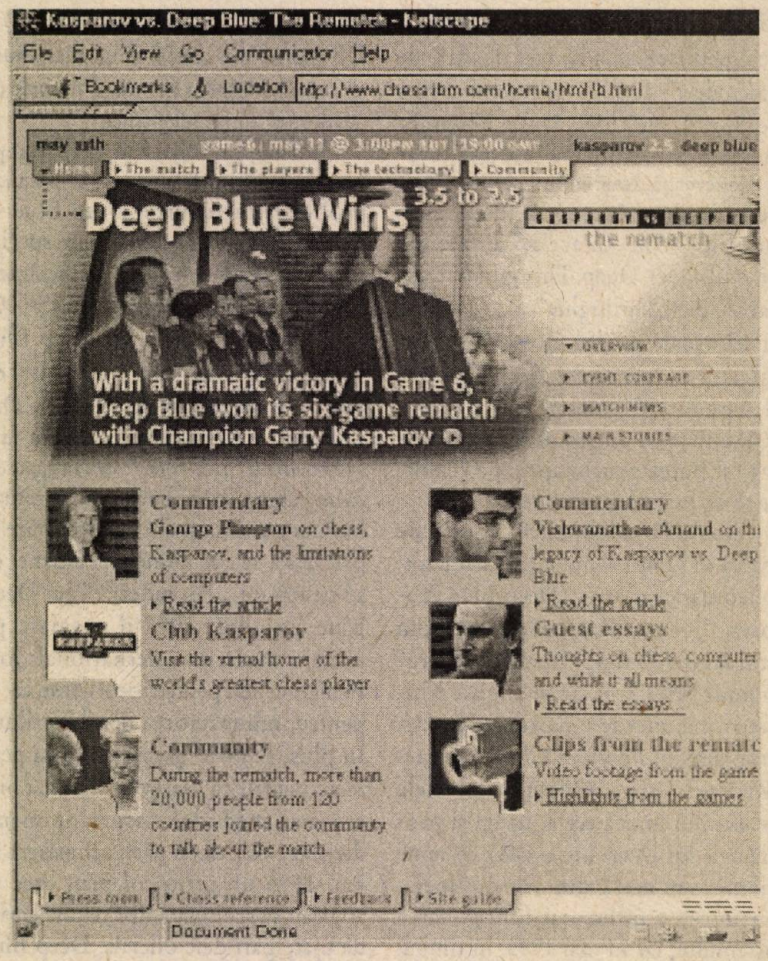
Din fericire (sau din păcate?), nu există (încă?) o definiție universal-acceptată a inteligenței. Cum ideea de „inteligență" se află la intersecția mai multor discipline (filosofie, psihologie și informatică, pentru a aminti doar câteva), nici nu e de mirare că soluția comună e greu de găsit. Dar mai mult, nici în cadrul unei discipline anume nu există consens în această problemă, iar în inteligența artificială situația pare de-a dreptul paradoxală: nici măcar IA-ul în sine nu are o definiție clară! În conformitate cu Russel și Norvig, a căror carte pare a fi noua „biblie" a domeniului, există cel puțin patru mari tabere în AI, fiecare dintre ele fiind caracterizată de două posibile opțiuni: **acțiune** versus **gândire**, și **uman** versus **rațional**. Concret, aproape toate definițiile AI-ului cad într-una din următoarele categorii:

- sisteme care **gândesc** într-un mod asemănător **omului**.
- sisteme care **acționează** într-un mod asemănător **omului**.
- sisteme care **gândesc** într-un mod **rațional**.
- sisteme care **acționează** într-un mod **rațional**.

Dacă pentru primele două categorii o definiție formală a inteligenței poate fi înlocuită cu teste standardizate (Testul Turing, respectiv Testul Turing Total), „sistemele ce **acționează rațional**" au dus la apariția conceptului de „agent", care este probabil cel mai... confuz concept din informatică! În fine, „sistemele ce **gândesc rațional**" se bazează pe formalismele logicii matematice și ridică două mari probleme: pe de o parte, reprezentarea cunoștințelor este extrem de dificil de formalizat (mai ales când apare factorul... incertitudine) iar pe de altă parte, constrângerile fizice (timp de lucru și memorie necesară) devin deseori prohibitive. Trăgând linia, observăm imediat că, în trei din cele patru accepțiuni, atât inteligența „naturală" cât și cea artificială rămân practic nedefinite, iar în al patrulea caz este limpede că IA-ul se află încă extrem de departe de ceea ce am putea numi

Ce s-a văzut pe Web?

Pe toată perioada meciului, IBM a întreținut un sit Web (www.chess.ibm.com) care a furnizat în permanență informații despre desfășurarea evenimentelor, comentarii ale specialiștilor în șah și în informatică, precum și o arhivă de documente relevante pentru acest eveniment (de la lecții de inițiere în șah până la eseuri și povestiri SF). Situl este încă accesibil și merită vizitat, atât pentru informațiile pe care le conține, cât și pentru exemplara sa organizare și înfățișare. Un detaliu: situl este găzduit de un computer IBM RS/6000, de același tip cu cel ce a constituit baza constructivă a lui Deep Blue.



„inteligență în condiții acceptabile".

Analiza de mai sus pare să ducă la o singură concluzie: în momentul de față, nu există o metrică unanim acceptată pe baza căreia să se poată decide dacă un sistem **este** (sau **poate fi**) inteligent. În acest context, întrebarea „este sau nu este Deep Blue inteligent?" este sortită să rămână momentan fără răspuns. Chiar și în aceste condiții, ar mai fi de adăugat un lucru extrem de important: în ultima instanță, faimosul răspuns va fi dat – poate – de o simplă nuanță semantică. Concret, o definiție de tipul „inteligența este o calitate a **ființei umane**..." exclude de la bun început posibilitatea ca Deep Blue să poată fi inteligent (dar, în mod analog, exclude și posibilitatea ca un extraterestru să poată fi inteligent!), pe când o formulare de genul „o **entitate** este inteligentă dacă..." ar permite atât „obiectelor" cât și ființelor extraterestre să fie inteligente.

Pentru a arăta că nu e vorba (doar) de un joc de cuvinte, voi menționa aici un excelent exemplu al acelorași Russel și Norvig: dacă acum câteva sute de ani „zborul" era doar de apanajul animalelor (obiectele și oamenii **nu** zburau pe atunci), în zilele noastre nimeni nu se scandalizează dacă-i spui că „Popescu a zburat la București" sau „avionul zboară repede". În schimb, o activitate extrem de similară zborului, **înotul**, a rămas rezervată ființelor (umane sau nu), obiectele mulțumindu-se să **navigheze** (de exemplu bărcile, submarinele și vapoarele). Mă întreb cum ar putea fi caracterizat oare un **robot** care se deplasează prin apă într-un mod care imită perfect omul: înoată sau navighează?

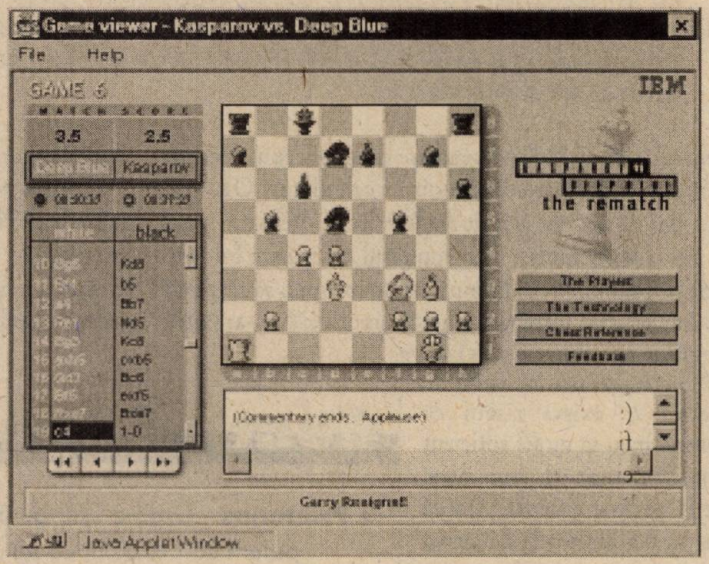
We are the champions

Chiar dacă faci parte dintre cei convinși că răspunsul la întrebarea „este Deep Blue inteligent?" este mai dificil decât întreg procesul ce a dus la apariția lui DB, există extrem de multe teze incitatoare care te pot face să te lansezi în interminabile discuții pe această temă. Spre exemplu, dacă accepți ideea ca „inteligența artificială se ocupă de soluționarea unor probleme pe care omul nu le poate rezolva decât folosindu-și inteligența", e clar că Deep Blue posedă o inteligență artificială. În schimb, dacă îi asculți pe experții care susțin că DB va face progrese în primul rând prin mărirea numărului de procesoare, memorarea mai multor deschideri și completarea tabelei de final-de-joc cu soluțiile tuturor finalurilor cu 6 și 7 piese, nu poți să te abții să zici: „Dar asta este forță brută și n-are nici o legătură cu inteligența!" De parcă soluția la sporirea inteligenței umane constă în „montarea" a două-trei creiere suplimentare și în memorarea tablei înmulțirii până la un milion!

În opinia mea, cea mai importantă lecție legată de fenomenul Deep Blue constă în faptul că omenirea este capabilă să creeze adevărate bijuterii tehnologice care păreau irealizabile cu câțiva ani în urmă. Nivelul de inteligență al lui Deep Blue este – cred eu – un aspect secundar în comparație cu faptul că această uimitoare mașină **există**. Sau, cu alte cuvinte, Deep Blue este un pas mic pentru inteligența artificială, dar un pas uriaș pentru tehnologia umană! ❑

Ultima mutare

Profesioniștii și pasionații șahului au avut posibilitatea să urmărească desfășurarea marii revanșe prin intermediul sitului Web special proiectat pentru acest eveniment (http://www.chess.ibm.com — situl este în continuare accesibil). Vizualizarea mutărilor unei partide se poate face prin mai multe metode, dintre care cea mai spectaculoasă este cea furnizată de un applet Java care prezintă grafic situația tablei de șah (într-o fereastră specială pot fi citite comentariile unor specialiști referitoare la fiecare mutare în parte). Captura de ecran de mai jos prezintă poziția din partida decisivă în care Kasparov (cu negrele, la mutare) s-a recunoscut învins.



Bibliografie

- David Gelernter. *How Hard is Chess?* TIME, 19 Mai 1997.
- Garry Kasparov. *IBM Owes Mankind a Rematch.* TIME, 26 Mai 1997.
- Drew McDermott. *How Intelligent is Deep Blue?* New York Times, 14 Mai 1997.
- Monty Newborn. *Kasparov versus DEEP BLUE: Computer Chess Comes Of Age.* Springer, 1997.
- Stuart Russel și Peter Norvig: *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 1995.

Ionuț Mușlea este doctorand la University Of Southern California din Los Angeles. E-mail: muslea@ISI.EDU.

Pentru preșcolari și pentru specialiști

LOGO – un LIMBAJ serios ?

de Mircea Sârbu

Probabil că ați auzit de LOGO. Un limbaj despre care se spune că a fost conceput pentru copii. Gândul ne duce involuntar la BASIC. Urmează de obicei o grimasă care combină expresia disprețului, a ororii și a plictiselii. Până când, într-o zi, ajungi să-l descoperi...

În general, limbajele de programare sunt concepute pentru a facilita comunicarea dintre om și calculator. Ok, veți spune, dar de ce sunt atât de multe? Aveți dreptate – căci sunt într-adevăr multe, doar numărul celor *încă* folosite este de ordinul miilor... Ei bine, nu-mi propun să răspund la această legitimă întrebare din două motive: (1) s-ar putea ca paginile acestei reviste să fie insuficiente pentru un astfel de demers; (2) există riscul ca răspunsul meu să nu fie suficient de fundamentat și, mai ales, atractiv pentru cititori. Așa că prefer să mă limitez la a încerca niște răspunsuri simple pentru niște întrebări simple: De ce a apărut limbajul Logo? Prin ce se deosebește Logo de alte limbaje? Ce se poate face cu Logo? De ce e bine să știți Logo?

De ce LOGO?

Răspunsul la această întrebare constituie prima surpriză furnizată de Logo și, totodată, un prim element care particularizează acest limbaj printre toate celelalte: Logo nu a fost conceput ca un instrument pentru profesioniștii calculatoarelor, ci pentru copii. Altfel spus, Logo provine dintr-un demers pedagogic, iar Harold Abelson ne-o spune într-un mod foarte concis: *„Logo este un nume pentru o filosofie a educației și pentru o familie în continuă evoluție de limbaje de programare care ajută la realizarea acesteia".*

Stați, nu abandonați încă articolul! Chiar dacă a fost conceput pentru copii, Logo este extrem de interesant, puternic și... simpatic. Este însă obligatoriu să amintim (pe scurt, desigur) câteva repere din istoria sa de peste 30 de ani.

Totul a pornit de la studiile unui psiholog elvețian, pe nume Jean Piaget, privind mecanismele învățării la copii. Rezultatele sale s-au constituit într-o nouă teorie a educației, numită *constructivism*, care a revoluționat știința pedagogică. Pe scurt, constructivismul consideră cunoașterea ca fiind produsul creației intelectuale a celui care învață, rezultând din interacțiunea acestuia cu oamenii și mediul înconjurător. Pare foarte natural, dar adesea practica pedagogică neglijează încă acest adevăr simplu: învățarea este un act de creație, nu de memorare. Transpunerea în practică a teoriei lui Piaget avea deci nevoie de instrumente specifice.

Pe la mijlocul anilor '60, un matematician care lucrase cu Piaget ajunge în Statele Unite. Și nu oriunde, ci tocmai la MIT (*Massachusetts Institute of Technology*), o adevărată Mecca a informaticii. Numele său: ***Seymour Papert***. La MIT, Papert înființează, împreună cu Marvin Minsky, Laboratorul de Inteligență Artificială, care devine curând incubatorul unor realizări excepționale. Aici, împreună cu o echipă de specialiști, Papert realizează în 1967 prima versiune de Logo, un limbaj de programare care își propune să fie un instrument didactic în sensul teoriei lui Piaget, un instrument care să stimuleze tocmai acea creație intelectuală care stă la baza învățării.

Logo se bazează pe un limbaj de programare – folosit intens în Inteligența Artificială – numit Lisp (programatorii

Cum e Logo?

- **Prietenos** – Logo e ușor de înțeles; putem să-l folosim ca pe un „ajutor al gândirii";
- **Extensibil** – Logo poate să învețe noi comenzi pe baza celor deja existente;
- **Tolerant** – Logo oferă un feed-back imediat prin mesaje informative și utile;
- **Flexibil** – Logo este la fel de bun pentru preșcolari și pentru studenți în anii terminali;
- **Puternic** – Logo oferă tot ce este necesar pentru a realiza programe oricât de complexe.

„Două idei fundamentale străbat această carte. Prima este că se pot proiecta calculatoarele în așa fel încât a învăța să comunici cu ele poate fi un proces natural, asemănător mai degrabă modului în care înveți limba franceză trăind în Franța decât încercând s-o înveți prin procesul nenatural de predare în clasă a limbilor străine. A doua este că a învăța să comunici cu calculatorul poate schimba modul în care înveți alte lucruri. [...] Învățăm acum să facem calculatoare cu care copiilor să le placă să comunice. Când această comunicare se petrece, copilul învață matematica ca pe un limbaj viu. Mai departe, comunicarea matematică și cea alfabetică sunt transformate din lucruri străine – și deci dificile, cum sunt pentru cei mai mulți copii – în lucruri naturale, deci ușoare. Ideea de a «vorbi matematică» cu un calculator poate fi generalizată într-o viziune a învățării matematicii în «Țara Matematicii», adică într-un context care este pentru învățarea matematicii ceea ce este Franța pentru învățarea limbii franceze."

- **Seymour Papert**, în cartea
Mindstorms: Children, Computers and Powerful Ideas.

profesioniștii încă îl consideră un *dialect* al acestuia). De la Lisp a moștenit Logo flexibilitatea și puterea calculului simbolic, care l-au recomandat pentru numeroase proiecte avansate în matematică, lingvistică, robotică (da, chiar așa!), muzică și alte domenii. Spre deosebire însă de Lisp, Logo a fost gândit ca un „instrument pentru învățare", deci trebuia să fie în primul rând accesibil neofiților (chiar și copiilor), să fie expresiv și, mai ales, simplu.

Restul e istorie. La sfârșitul anilor '70 apar calculatoarele personale și primele implementări Logo pentru acestea (mai întâi Apple II și Texas Instruments TI 99/4). Răspândirea PC-urilor în deceniul următor va aduce cu sine utilizarea limbajului pe scară largă în multe școli din întreaga lume.

Un prim contact

Dacă am reușit să vă trezesc interesul, atunci cu siguranță sunteți nerăbdători să trecem la treabă. Desigur, ar trebui să citim înainte de toate o carte de programare în Logo, să facem niște exerciții pe hârtie... Dar cine are vreme de toate astea? Dacă-i așa de simplu, să intrăm direct în subiect. Lansăm un mediu Logo (de pildă Berkeley Logo) și vom vedea pe ecran un *promter* (în acest caz este semnul „?") care ne invită să introducem comenzi. Ce comenzi? Încă nu știm... Să încerc să mă prezint:

```
? mircea
I don't know how to mircea
```

Interesant... Adică: „nu știu cum să mircea" (sau, dacă vreți, „nu știu cum să *mircesc*"). Pesemne numele meu nu are nici o semnificație pentru Logo. Nu-i de mirare... Numerele și operațiile aritmetice sunt cu siguranță universale:

```
? 23 + 12
You don't say what to do with 35
```

Hop! Așadar operația a făcut-o, dar nu știe ce să facă cu rezultatul. Intuiția ne spune că ar trebui să plasăm în față o comandă. De exemplu să afișeze rezultatul. Să încercăm deci ceva de genul următor:

```
? print 23 + 12
35
```

Corect. Ideea pare să fie că Logo acceptă comenzi (de pildă `print`) urmate de argumente (în cazul nostru `35`). Argumentele însă pot fi și ele rezultatul unor operații (în exemplul de mai sus a fost vorba de o adunare: `23 + 12`). Vom avansa mai repede dacă veți încerca să urmăriți conversația de mai jos (numerotarea este doar pentru a ajuta discuția care urmează):

```
(1) ? print logo
    I don't know how to logo
(2) ? print "logo
    logo
(3) ? print first "logo
    l
(4) ? print "logo "e "simplu
    You don't say what to do with e
(5) ? print [logo e simplu]
    logo e simplu
(6) ? pr first [[limbajul logo] e simplu]
    limbajul logo
(7) ? show butfirst lput list "ca [buna ziua] [logo e
simplu]
    [e simplu [ca [buna ziua]]
```

Linia (1) este oarecum surprinzătoare: ne-am fi așteptat ca Logo să afișeze cuvântul „logo". Din mesaj deducem însă că a încercat să execute instrucțiunea (mai precis: procedura) „logo" și că aceasta nu există (adică nu e definită). În schimb, linia (2) produce rezultatul așteptat. Deci ghilimele din fața lui „logo" vor să spună că ceea ce urmează este un literal (deci un șir de caractere).

O primă concluzie: Logo încearcă să evalueze toate cuvintele. Ghilimele inhibă evaluarea.

Acum, linia (3) este mai ușor de înțeles. Bănuim că `first` este o procedură care primește ca argument cuvântul „logo". Nu apare nici un mesaj de eroare, deci instrucțiunea e validă. Nu-i greu să deducem că `first` extrage

prima literă a argumentului și o pasează instrucțiunii `print`, care o afișează.

O a doua concluzie: instrucțiunile Logo sunt, de fapt, proceduri. Ele pot produce ca rezultat fie anumite valori, fie realizarea unor acțiuni. Primele se numesc **funcții** iar celelalte **comenzi**. Este evident că `first` este o funcție iar `print` este o comandă.

Linia (4) ne înspăimântă la prima vedere: comanda `print` nu acceptă decât un singur argument! Ba chiar mai mult: o greșeală minoră (furnizarea mai multor argumente) este fatală! Liniștiți-vă, nu-i chiar așa. Comanda `print` (ca de altfel multe alte proceduri) pot accepta un număr variabil de argumente dacă se folosește o convenție de scriere:

```
(4') (print "logo "e "simplu)
     logo e simplu
```

În privința „toleranței", depinde de implementare. Berkeley Logo semnalează eroare de îndată ce a depistat-o. În schimb, SuperLogo (produs de Longman Logotron) execută întâi tot ce se poate și abia apoi furnizează mesajul de eroare (deci linia 4 va afișa întâi „logo" după care va da mesajul de eroare).

În linia (5) facem cunoștință cu un element foarte important în Logo: listele. O listă este cuprinsă între paranteze drepte și (în mod normal) nu se evaluează. Ce pot cuprinde listele? Elemente. Elementele pot fi cuvinte sau liste. Se observă caracterul recursiv al definiției listei. O listă întrucâtva specială este lista vidă, notată [].

Linia (6) relevă caracterul oarecum polimorfic al funcției `first`: dacă este aplicată unui cuvânt (ca în linia 3) produce (altfel spus, *returnează*) prima literă, dar dacă este aplicată unei liste, întoarce primul element al listei. Tot aici remarcăm forma scurtă a comenzii `print` (`pr`) și putem face observația că multe comenzi au astfel de forme, utile în regim conversațional. De asemenea, observăm un prim exemplu în care un element al unei liste este o listă.

Linia (7) e puțin mai complicată. Pentru ca să ne putem concentra asupra esențialului, să elucidăm întâi cuvintele necunoscute:

`show` — este similară cu `print`, cu singura diferență că, în cazul listelor, afișează și parantezele drepte (altfel spus: afișează lista, nu doar conținutul ei).

`butfirst` — e un fel de „inversă" a lui `first`: dacă argumentul e o listă, returnează lista fără primul element. Se poate aplica și asupra cuvintelor. Formă scurtă: `bf`.

`lput` — returnează lista furnizată de al doilea argument la care se atașează (ca ultim element) primul argument. Funcția „simetrică" este `fput` (elementul nou este adăugat în față).

`list` — construiește o listă din argumentele pe care le primește. În mod normal primește două argumente, dar dispune și de forma specială (cu paranteze) care admite oricâte argumente.

Să respirăm adânc... și să introducem un subtitlu:

Mecanismul evaluării

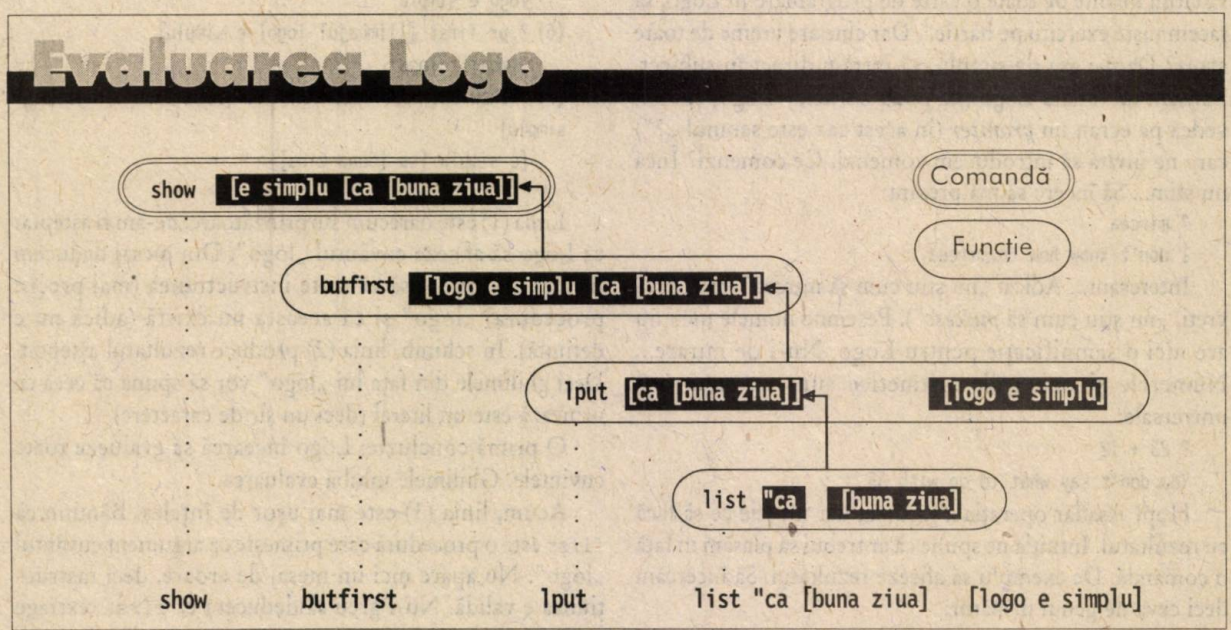
Ok. Să încercăm să urmărim ce se întâmplă:

Comanda `show` vrea să-și afișeze argumentul. Caută în dreapta sa și găsește funcția `butfirst`. Aceasta înseamnă că va trebui să afișeze ceea ce returnează `butfirst`, deci îi cere rezultatul. Se declanșează astfel procesul de evaluare a funcției `butfirst`.

Funcția `butfirst` așteaptă la rândul ei un singur argument (un cuvânt sau o listă). Îl caută în dreapta sa și găsește o altă funcției, `lput`. „Evaluează-te!" — îi ordonă, după care așteaptă rezultatul.

Aici lucrurile se complică puțin: `lput` vrea două argumente. În privința primului, lucrurile sunt simple: trebuie să fie imediat în dreapta sa. Acolo găsește funcția `list`, care va trebui să-i furnizeze prin evaluare primul argument (va fi cu siguranță o listă).

Dar unde este al doilea argument? Există două modalități de a-l găsi: (a) se *parsează* în continuare linia, pentru



a „sări" peste funcția `list` și cele două argumente ale sale; (b) se lansează evaluarea funcției `list`, care-și va „înghiți" argumentele necesare și va lăsa funcției `lput` al doilea argument. În cazul nostru ambele variante pot funcționa și este greu de spus pe care o aplică interpretorul. Pentru a discuta la obiect, să ne imaginăm că se aplică metoda (b).

Așadar, `list` așteaptă două argumente. Primul este (în acest caz) un cuvânt (`"ca`) iar al doilea o listă (`[buna ziua]`). E simplu, funcția `list` va produce ca rezultat lista: `[ca [buna ziua]]`.

Funcția `lput` primește deci această listă ca prim argument. Al doilea argument este ce a mai rămas după ce `list` și-a consumat argumentele: lista `[logo e simplu]`.

Acum, funcția `lput` poate să-și îndeplinească misiunea. Atașează ca ultim element al listei primite ca al doilea argument ceea ce i s-a furnizat ca prim argument. Rezultatul este, evident, lista `[logo e simplu [ca [buna ziua]]]`.

În sfârșit, `butfirst` poate să-și facă datoria. Va înmâna, deci, comenzii `show` lista pe care a primit-o ca argument, din care va îndepărta în prealabil primul element. `Show` nu face decât să pună pe ecran această listă: `[e simplu [ca [buna ziua]]]`.

Figura „Evaluarea Logo" încearcă să ilustreze acest mecanism.

Să sistematizăm puțin observațiile de până acum:

♦ Logo este un limbaj destinat interpretării. Interpretarea este un proces care se desfășoară în două etape: *parsarea* și *evaluarea*. Parsarea constă în identificarea și separarea elementelor de limbaj în timp ce evaluarea constă în execuția propriu-zisă a procedurilor.

♦ Evaluarea este inițiată de la stânga spre dreapta, dar se realizează de fapt de la dreapta spre stânga. Este de reținut că o procedură nu poate fi executată decât după ce au fost evaluate toate argumentele sale.

♦ În regim conversațional (modul uzual de utilizare a interpretorului Logo) prima procedură trebuie să fie o comandă.

Nu pretind că aceste enunțuri sunt riguroase (ca de altfel și multe altele din acest articol). Ele pot însă constitui un punct de pornire în învățarea acestui limbaj care, la prima vedere, pare oarecum „excentric".

Variabile

Să continuăm conversația pe care am analizat-o până acum cu câteva elemente noi:

```
(8)   ? make "var "valoare
(9)   ? pr "var
      var
(10)  ? pr thing "var
      valoare
(11)  ? make "valoare 154
(12)  ? pr :valoare
      154
(13)  ? pr thing :var
      154
```

În linia (8), procedura `make` primește două argumente. Deoarece interpretorul Logo nu emite nici un protest și nu face altceva decât să afișeze din nou promterul, bănuim că procedura e o comandă și că ea s-a executat cu succes, dar efectul ei nu este vizibil pe ecran.

Linia (9) are doar rolul de a pregăti spectacolul care urmează. Este o comandă absolut banală.

În fine, linia (10) ne dezvăluie efectul ascuns al comenzii `make`: așa cum bănuiați, cuvântul `"var` desemnează acum un anumit obiect, în cazul de față cuvântul `"valoare`. Altfel spus, am făcut (*to make*) din `"var` o variabilă, deci un fel de cutiuță în care punem o valoare. Să remarcăm însă faptul că literalul `"var` a rămas în continuare un simplu șir de caractere (vezi linia (9)). Este un nume pentru un alt obiect.

Procedura `thing` este o funcție care returnează valoarea unei variabile al cărei nume este dat ca parametru (dacă numele nu desemnează un obiect, `thing` emite un mesaj de eroare).

În linia (11) denumim prin cuvântul `"valoare` numărul 154. Un număr nu este altceva decât un șir de caractere care, din întâmplare, sunt cifre (nu doar...). Interpretorul Logo tratează numerele ca și când ar avea ghilimele în față. În linia (12) ne convingem că am obținut efectul scontat și, în plus, introducem o altă notație pentru valoarea unei variabile (caracterul „:" plasat înaintea unui cuvânt este echivalent cu aplicarea funcției `thing` asupra numelui respectiv prefixat de ghilimele).

Linia (13) merită o atenție specială. Evaluarea expresiei `:var` produce literalul `"valoare`. Acesta devine argumentul funcției `thing`. Deoarece `"valoare` este numele unei valori (în speță numărul `154`), funcția `thing` va furniza comenzii acest număr. Observăm că valoarea unei variabile poate fi la rândul ei o variabilă (care, la rândul ei, poate avea ca valoare numele unei alte variabile etc.).

Încă o observație legată de linia (13). O exprimare perfect echivalentă este:

```
? pr thing thing "var
```

care, desigur, produce același rezultat. Ne-am putea aștepta ca expresia `::var` să fie și ea validă, dar nu este.

Pe scurt: variabilele Logo nu sunt nimic altceva decât niște nume date unor valori. Aceste valori pot fi cuvinte (inclusiv numere) sau liste, ceea ce înseamnă că variabilele nu sunt tipizate.

Predicate

Păstrând contextul pe care l-am definit vorbind despre variabile, ar fi util să mai introducem câteva noțiuni a căror importanță se va dezvălui ceva mai târziu. Așadar, să continuăm conversația.

```
(14)  ? pr wordp :var
      true
(15)  ? pr first namep "var
      t
(16)  ? pr not namep thing :var
      true
(17)  ? if greaterp 157 thing :var [pr [mai mare]]
      mai mare
```

```
(18)  ? ifelse 153 > thing :var [pr ">][pr "<=]
      <=
(19)  make "de.facut [pr bf [say hello, world!]]
(20)  run :de.facut
      hello, world!
```

În linia (14) apare o procedură ceva mai deosebită, `wordp`. Este clar că e o funcție, deoarece returnează comenzii `print` o valoare spre afișare. Dacă ne punem intuiția la lucru, vom observa că funcția se aplică de fapt *cuvântului* `"valoare`, iar numele ei provine, după toate aparențele, din englezescul *word* (care înseamnă „cuvânt"). După ce vedem și rezultatul tipărit, `true` (adică „adevărat"), bănuim deja că misterioasa funcție determină dacă argumentul pe care-l primește este sau nu este un cuvânt (adică un *literal*).

Într-adevăr, așa este. De unde provine terminația „p" a numelui funcției? De la faptul că funcția este un predicat logic, adică o funcție care stabilește dacă o afirmație (în cazul acesta faptul că argumentul este un literal) este adevarată (`true`) sau falsă (`false`). Logo este dotat cu multe predicate, al căror nume se termină de regulă cu litera „p" (câteva excepții vor fi semnalate ceva mai jos). Atenție: nu toate procedurile terminate în „p" sunt predicate!

Linia (15) vine să demistifice (dacă e cazul) caracterul acestor două valori „speciale", `true` și `false`. Predicatul `namep` determină dacă argumentul pe care-l primește este numele unei variabile. În acest caz răspunsul este afirmativ (vezi linia 8) și deci returnează valoarea `true`, pe care – surpriză – funcția `first` o tratează fără nici o problemă, returnând prima literă! Așadar, `true` și `false` nu sunt deloc speciale: sunt simple literale.

Linia (16) nu produce nici o revelație: într-adevăr, funcția `namep` funcționează corect și, dacă e cazul, poate să producă și valoarea `false`. Însă operatorul logic `not` schimbă desigur rezultatul în `true`. Iată deci o primă excepție de la regula sufixului „p": operatorii logici `and`, `or` și `not` sunt funcții care returnează doar `true` sau `false`, deci sunt predicate logice, dar nu se termină în „p".

Linia (17) este cea care relevă importanța predicatelor: prin intermediul lor poate fi controlat fluxul execuției. Comanda `if` primește două argumente, dintre care primul trebuie să fie un predicat (desigur, se poate plasa aici și literalul `true` sau `false`, dar în cazul acesta ce rost are să mai testăm?). Al doilea argument este o listă de instrucțiuni, care se execută în cazul în care evaluarea primului argument produce valoarea `true`. În caz contrar, comanda nu produce nici un efect.

Predicatul `greaterp` este o comparație aritmetică simplă. Deoarece evluarea celui de al doilea argument produce valoarea `154` (vezi linia 13), predicatul se evaluază la `true`, comanda `if` va lansa în execuție lista de instrucțiuni pe care a primit-o ca al doilea argument.

Linia (18) aduce două elemente noi. Comanda `ifelse` este, ați ghicit, un test cu două ieșiri, primind în consecință și o a doua listă de instrucțiuni, care va fi executată dacă predicatul produce valoarea `false`. Mai interesant este însă predicatul: este vorba despre același `greaterp`, însă exprimat într-o formă mai naturală (forma *infixată*).

Ultimele două linii nu vor decât să dezvăluie faptul că în spatele structurilor de control din Logo (printre care se numără și `if`, `ifelse`, `repeat` și multe altele) nu e nici o magie, ci doar instrucțiunea `run` (care, ciudat lucru, poate fi considerată fie funcție, fie comandă, după cum execuția listei de instrucțiuni produce sau nu un rezultat). Ceea ce este însă demn de reținut este faptul că în Logo nu există nici o diferență între date și instrucțiuni! Altfel spus: instrucțiunile pot fi date pentru alte instrucțiuni, datele pot fi de fapt instrucțiuni care acționează asupra altor date.

Această caracteristică (moștenită din Lisp) este o consecință a faptului că Logo este un limbaj interpretat (în cazul limbajelor compilate, cum este Pascal, codul sursă este transformat în cod mașină înainte de execuție, deci programul compilat nu mai are acces la propria sa sursă). Interesant este de semnalat că identitatea structurală a instrucțiunilor cu datele se regăsește și la nivelul programelor în cod mașină (lucru pe care cei care lucrează în limbaje de asamblare îl pot confirma).

Îmbogățind limbajul

În mod normal, o abordare sistematică ar trebui să continue cu prezentarea structurilor de date, cu clasificarea comenzilor și funcțiilor etc. Dar pentru că nu mi-am propus o abordare sistematică și riguroasă, ci doar o introducere rapidă și atractivă, mă grăbesc să ajung la partea cea mai „fierbinte" a prezentării, cea pe care sunt sigur că o așteptați deja...

Bănuiesc că întrebarea care vă stă pe buze sună cam așa: „Oricâte proceduri știe Logo, este imposibil ca ele, doar ele, să poată să-mi rezolve problemele mele concrete. Am eu posibilitatea să-mi definesc propriile mele proceduri, să le denumesc cum vreau, să le memorez și să le folosesc oricând cred eu de cuviință?". Răspunsul este categoric afirmativ.

Iată un exemplu absolut banal:

```
(1) to salut
(2) print [Hello, world!]
(3) end
```

Am definit o procedură (în acest caz este o comandă) care, atunci când este invocată, afișează mesajul `Hello, world!`. Înainte de a ne ocupa mai îndeaproape de semnificația liniilor de mai sus, trebuie să menționez că definirea procedurilor nu se face în mod conversațional. Este vorba despre o editare de text similară cu cea a surselor pentru alte limbaje de programare. Diverse interpretoare Logo dispun de editoare specializate sau permit lansarea unui editor de text la alegere (de pildă cel din Norton Commander). De cele mai multe ori, editorul este lansat prin comanda `edit` urmată de numele procedurii pe care vreți să o editați. Oricum, nu strică să citiți capitolul relevant din manualul de utilizare.

Să revenim la exemplul dat. Linia (1) repezintă antetul definiției, constând din comanda `to`, urmată de numele pe care îl dăm procedurii (în cazul nostru `salut`) și, dacă e cazul, de parametrii formali ai procedurii.

Ultima linie a definiției conține, obligatoriu, doar cuvântul `end`. Între prima și ultima linie se desfășoară o succesiune de instrucțiuni, care constituie corpul definiției. În cazul nostru, corpul definiției este format dintr-o singură linie (2), cuprinzând o singură comandă.

De îndată ce definiția a fost încheiată, ea poate fi utilizată fie în regim conversațional, fie pentru a defini alte proceduri. Procedurile, împreună cu alte elemente pe care le-am definit (de pildă variabilele), formează *contextul* sesiunii, sau spațiul de lucru (*workspace*). Acesta poate fi salvat (prin comanda `save`) și restaurat apoi într-o altă sesiune de lucru (prin comanda `load`).

Desigur, procedurile nu prea sunt utile fără argumente. Să definim deci o procedură cu un argument:

```
(1) to inc :in
(2) output :in + 1
(3) end
```

Argumentul este o variabilă locală procedurii (adică: vizibilă doar pentru instrucțiuni din corpul acesteia) care se declară în cadrul comenzii `to`, imediat după numele procedurii. Numele pe care i-l atribuim este la alegerea noastră (în acest caz este `in` – remarcați convenția prin care numele trebuie precedat de caracterul „:") iar valoarea cu care este inițializată este cea pe care o primește la apel:

```
(21) ? pr inc 23
     24
```

Procedura `output` (în forma scurtă `op`) are ca efect încheierea execuției procedurii în care apare și returnarea către interpretor a valorii argumentului său – în cazul de mai sus `23 + 1`, deci `24`). Este interesant de remarcat faptul că `output` este o comandă (deci nu poate să apară ca argument al unei proceduri), dar procedura în care apare este o funcție. O comandă înrudită cu `output` este `stop`, care încheie execuția procedurii în care apare, dar nu determină returnarea unei valori (este folosită deci la definirea altor comenzi).

Ar fi interesant să încercăm să modificăm definiția funcției `inc` astfel încât să accepte un eventual al doilea argument care să furnizeze o valoare, diferită de 1, care să fie adunată la valoarea primului argument. E simplu:

```
(1) to inc :in [:pas 1]
(2)  output :in + :pas
(3) end
```

Prin faptul că a fost scris între paranteze drepte s-a menționat că `:pas` este un argument opțional. În aceeași paranteză se specifică valoarea pe care acesta o primește în cazul în care nu este furnizată la apel (aici poate să apară o întreagă instrucțiune – în care pot fi implicate argumentele anterior declarate – care se evaluează la lansarea în execuție a procedurii).

Apelarea procedurii în această formă se face la fel ca și în cazul precedent dacă se furnizează un singur argument. Altfel, este obligatorie încadrarea apelului între paranteze rotunde:

```
(22) pr (inc 23 4)
     27
```

Bibliografie

Cărți în limba română:

♦ **Ion Diamandi**: *Cine știe LOGO? (Introducere în limbajul Logo)*, editura Agni, 1994.
Cartea se adresează elevilor si profesorilor din clasele gimnaziale și prezintă atât elemente de operare (pentru interpretoarele Logo de pe calculatoare HC - Spectrum și PC - DOS) cât și noțiuni de programare. Structurată ca un manual, prezintă numeroase exemple, propune probleme și activități practice, iar în final oferă o referință a limbajului. Recomandată pentru inițiere, indiferent de vârstă.

♦ **Petru Dumitriu**: *LOGO – 16 lecții și aplicații*, editura SPOT 1996.
Este tot o carte de inițiere, dar este mai degrabă orientată spre învățământul primar.

♦ **Petru Dumitriu**: *LOGO – 500 de proceduri*, editura SPOT 1997.
Cartea abordează prin intermediul limbajului Logo un domeniu fascinant: fractalii. Recursivitatea este intens folosită, figurile generate sunt adesea spectaculoase, iar abordarea este intuitivă.

Cărți în limba engleză:

♦ **Brian Harvey**: *Computer Science Logo Style, 2/e*, MIT Press, 1997.
Este cartea de referință cea mai larg răspândită. Este structurată în trei volume (I – *Symbolic Computing;* II – *Advanced Techniques;* III — *Beyond Programming*) și acoperă practic toate aspectele legate de limbajul Logo. Prima ediție (în două volume) a apărut tot la MIT Press, în 1985. Autorul, Brian Harvey, este cel care a realizat și interpretorul freeware *Berkeley Logo*.

♦ **Seymour Papert**: *Mindstorms - Children, Computers, and Powerful Ideas*, Basic Books, 1980
Este o carte deja „clasică" în privința impactului informaticii în educație. Nu este o carte de programare.

♦ **George Lukas, Joan Lukas**: *Logo: Principles, Programming & Projects*, Brooks/Cole, 1986

♦ **Pat Richardson, Judy Dobson**: *Problem Solving with Logo*, Addison-Wesley, 1984

♦ **Donna Bearden**: *A Bit of Logo Magic*, Reston Publishing, 1985

♦ **Boris Allan**: *Introducing Logo*, London Granada, 1984

Logo permite definirea de proceduri având orice număr de argumente (precizat sau nu). Iată cum se poate modifica funcția `inc` astfel încât să poată primi oricâte argumente care se adună la primul argument:

```
(1) to inc :in [:pas "1] [:rest]
(2) local "sum make "sum :in + :pas
(3) local "i make "i 1
(4) repeat count :rest ~
(5)     [make "sum :sum + item :i :rest ~
(6)      make "i :i + 1]
(7) op :sum
(8) end
```

Definiția a devenit mai complicată, dar a rămas inteligibilă. Câteva precizări:

♦ A apărut un al treilea argument opțional, `rest`. Spre deosebire de `pas`, acesta nu are o valoare implicită. În această situație (fiind și ultimul dintre argumentele definite) acesta va fi obligatoriu o listă care va „culege" toate argumentele suplimentare (deci în plus față de cel obligatoriu și cel opțional) furnizate la apelul funcției în forma cu paranteze rotunde. Astfel, în cazul apelului:

```
(23) ? pr (inc 10 20 30 40 9)
     109
```

variabila locală `rest` va primi la apel ca valoare lista `[30 40 9]`.

♦ Comanda `local` declară o variabilă locală procedurii. Scopul este de a evita interferența cu variabile globale care ar putea să aibă același nume. Atenție: comanda doar *declară* variabila ca fiind locală, dar nu-i atribuie nici o valoare, pentru această sarcină fiind folosită comanda `make`.

♦ Funcția `count` furnizează numărul de elemente al listei pe care o primește ca argument (de fapt, știe să facă mai multe, dar toate la vremea lor).

♦ Misteriosul caracter special „~" (tilda) nu face altceva decât să precizeze că linia pe care o încheie se continuă pe linia următoare.

De aici încolo e simplu: comanda `repeat` va executa lista de instrucțiuni pe care o primește ca al doilea argument de un număr de ori specificat de primul argument. Funcția `item` furnizează, așa cum ne așteptam, al `i`-lea element al listei indicate prin al doilea argument (în acest caz, `rest`).

Să remarcăm că singura diferență care a mai rămas între funcția noastră `inc` și funcția `sum` oferită de Logo este faptul că `inc` poate fi apelată și cu un singur argument (caz în care îl incrementează cu o unitate). Bănuiesc că unii dintre cititori se gândesc acum că am putea să definim și singuri multe dintre procedurile pe care ni le oferă Logo. Așa și este, iar între procedurile definite de noi și cele „native" interpretorul nu face nici o discriminare. Mai mult, unele dintre procedurile oferite de Logo sunt de fapt definite exact ca cele pe care le definim noi, ca simpli utilizatori. Acestea sunt însă depuse într-o bibliotecă specială, care este încărcată de interpretor în mod implicit. Spre deosebire de acestea (numite proceduri de bibliotecă - *library procedures*), majoritatea procedurilor native sunt implementate în interpretor (caz în care se numesc primitive) pentru a furniza performanțe superioare (prin evitarea parsării și prin diverse optimizări).

O idee de utilizare a procedurilor ar putea fi definirea unor proceduri echivalente cu cele predefinite, dar care să poarte numele pe care le dorim noi (e relativ simplu: se jonglează puțin cu argumentele și se apelează primitiva vizată). Inutil. Cei care au proiectat Logo au prevăzut o astfel de comandă:

```
(23) ? copydef "scrie "print
(24) ? scrie "Salut!
     Salut!
```

Structuri de date

Merită, cred să revenim puțin asupra structurilor de date din Logo. Am menționat deja că variabilele Logo nu sunt tipizate, dar s-ar putea să credeți că, totuși, cuvintele (printre care și numerele) și listele oferă prea puține posibilități pentru a reprezenta structuri de date complexe.

În primul rând trebuie să arăt că listele se pretează foarte bine atât la reprezentarea structurilor liniare cât și a celor arborescente. Iată două exemple:

1. `[Vasile Barbu 37 M [Ploiesti] [Fictiv srl]]`
2. `[Bogdan [Alex [Vasile Maria] Ioana [Valentin Valeria]]]`

În primul exemplu este vorba de o structură liniară, de tip „articol" (*record*), în timp ce în al doilea este vorba despre o structură arborescentă (mai precis: este un arbore genealogic, reprezentat ca arbore binar pe trei niveluri).

Desigur, aceste construcții pot fi îmbinate în cele mai diverse moduri. Dacă mai adăugăm și abilitatea lui Logo de a manevra fișiere, posibilitățile de reprezentare a structurilor de date nu mai par atât de sărace.

Este demn de semnalat că este relativ ușor de a construi (un fel de) tipuri de date, înțelegând prin aceasta structuri de date însoțite de seturi de operații asociate acestora (pentru o tratare riguroasă a subiectului, vezi articolul lui Mihai Budiu „Tipuri de date"). De pildă, stiva (*stack*) și coada (*queue*) sunt furnizate de Logo, prin intermediul operațiilor specifice (`push` și `pop` pentru stiva, `queue` și `dequeue` pentru coadă) care sunt funcții de bibliotecă predefinite.

O structură importantă, despre care nu am vorbit până acum, este *tabloul* (`array`). Acesta este asemănător cu o listă, dar are un număr fix de elemente. Iată un exemplu:

```
(25)   make "t {1 2 alfa beta [a b c] {98 99}}
(26)   pr :t
       {1 2 alfa beta [a b c] {98 99}}
(27)   show item 5 :t
       [a b c]
(28)   pr first :t
       1
(29)   setitem 5 :t bf item 5 :t
(30)   pr :t
       {1 2 alfa beta [b c] {98 99}}
(31)   show arraytolist :t
       [1 2 alfa beta [a b c] {98 99}]
```

Variabila `t` este un tablou de lungime 5 (adică are 5 elemente). După cum se vede în lina (25), elementele tablo-

ului pot fi orice tip de date, inclusiv alte tablouri. Spre deosebire de liste, tablourile se scriu între acolade, așa cum se vede și în linia (26).

Funcția `item` extrage un element din tablou, pe baza unui index (de altfel este aplicabilă și listelor, așa cum am văzut deja). Interesant este că funcția `first` este aplicabilă unui tablou, dar funcțiile înrudite (`butfirst`, `last`, etc.) nu sunt.

Comanda `setitem` modifică un element al tabloului, iar funcția `arraytolist` produce o listă având elemente identice cu cele ale tabloului.

Este interesant de remarcat că funcția `listtoarray` este o cale foarte comodă de a crea un tablou, fără a recurge la o variabilă. Funcția „normală" care definește un tablou, `array`, este incomod de folosit pentru că produce un tablou care are toate elementele [] (lista vidă) și deci trebuie folosită comanda `setitem` pentru a plasa valori. În schimb, `array` are avantajul că permite declararea explicită a dimensiunii și, eventual, a originii, adică indicele primului element (implicit este 1). De pildă:

```
(32)  make "x array 5 3
(33)  setitem 3 :x "alfa
```

Indicele folosit în linia (33) coincide cu originea, deci se referă la prima poziție a tabloului. Așadar, valoarea variabilei `x` devine `{alfa [] [] [] []}`.

În fine, mai trebuie remarcat că deoarece elementele unui tablou pot fi tablouri, este ușor de dedus că se poate lucra cu tablouri multidimensionale. Funcția pentru crearea acestora, `mdarray` (procedură de bibliotecă), primește ca prim parametru o listă de valori care specifică „dimensiunile dimensiunilor" (numărul elementelor pentru fiecare dimensiune). Evident, numărul dimensiunilor este chiar numărul elementelor din listă. Al doilea parametru este opțional și specifică originea. De altfel, cititorii pe care Logo i-a „prins" deja, pot încerca o definiție alternativă pentru `mdarray`, ca exercițiu.

Proprietăți

Dacă, într-o anumită măsură, structurile de date prezentate până acum vă sunt familiare, listele de proprietăți reprezintă o noutate (cel puțin pentru cei ce nu știu nimic despre Lisp). Așadar, încercați să urmăriți conversația de mai jos:

```
(34)  pprop "patrat "poz [0 0]
(35)  pprop "patrat "lat 100
(36)  make "negru [000 000 000]
(37)  pprop "patrat "col :negru
(38)  pr gprop "patrat "lat
      100
(39)  show gprop "patrat "aria
      []
(40)  make "patrat [lin 100]
(41)  show se :patrat plist "patrat
       [lin 1 col [000 000 000] lat 100 poz [0 0]]
```

Linia (34) ilustrează utilizarea comenzii `pprop`, care în această situație are un rol triplu: 1) creează lista de pro-

Resurse Web

General:

♦ **Logo Foundation** (http://el.www.media.mit.edu/groups/logo-foundation/)
The Logo Foundation este o organizație educațională non-profit a cărei scop este să ofere informații despre Logo și să ofere sprijin celor interesați în utilizarea softurilor și a mediilor educaționale bazate pe Logo. Situl Web cuprinde, printre altele, arhivele revistei *Logo Update* (editată de trei ori pe an de Logo Foundation), informații despre grupurile de utilizatori Logo din întreaga lume, despre listele de discuții pe tema Logo din Internet, despre seminarii și conferințe, cărți, articole, software și în general tot ce este de interes în lumea Logo. Este, fără dubii, punctul de start ideal către alte resurse Web. Tot de aici există trimiteri la adresele Internet de la care se pot descărca software gratuit (în principal UCBLogo și MSWLogo).

♦ **EUROLOGO Web Server** (http://www.eurologo.org/)
Cuprinde articole și noutăți privind activitatea mișcării Logo din Europa. Diverse legături spre situri de interes.

Software:

♦ **Berkeley Logo download** (ftp://anarres.cs.berkeley.edu/pub/ucblogo/)
Este interpretorul gratuit dezvoltat de Brian Harvey la *University of California at Berkeley* - de unde-i și provine numele: UCBLogo. Sunt disponibile la această adresă distribuții complete pentru DOS (blogo.exe - 336 KB), Macintosh (ucblogo.sea.hqx) și Unix (ucblogo.tar.Z). O versiune ASCII a manualului de utilizare găsiți la http://http.cs.berkeley.edu/~bh/usermanual.

♦ **MSWLogo download** (http://www.softronix.com/logo.html)
Este o versiune pentru Windows dezvoltată de George Mills pe baza nucleului Berkeley Logo, pe care l-a extins cu elemente multimedia și de comunicații, precum și cu posibilități de a realiza aplicații Windows interactive. Există versiuni pe 16 biți pentru Windows 3.x (arhiva: 1,2 MB) și pe 32 de biți pentru Windows 95/NT (arhiva: 1,4 MB). Cuprinde help și exemple. Este disponibilă și o versiune în format Word a documentației (coincide cu help-ul – cuprinde integral documentația pentru UCBLogo).

♦ **MicroWorlds 2.0 - demo** (http://www.lcsi.ca/getdemos.htm)
Este un software comercial dezvoltat de LCSI (*Logo Computer Systems, Inc.*), disponibil pentru Windows 95 și pentru Macintosh. Versiunile demonstrative nu salvează spațiul de lucru și nu au help, în rest sunt perfect funcționale.

prietăți numită patrat; 2) adaugă proprietatea numită poz; 3) atribuie acestei proprietăți valoarea [0 0].

Primul rol menționat este accidental: creează lista doar dacă aceasta nu există deja. Astfel, comenzile pprop din liniile (35) și (37) nu re-creează lista de proprietăți patrat, ci doar îi adaugă proprietățile lat (lungimea laturii) și col (culoarea), cu valorile corespunzătoare.

Constatăm că o listă de proprietăți seamănă oarecum cu o variabilă, în sensul că are un nume prin care este referită. Dar, spre deosebire de variabile, nu are o valoare „directă" (deci accesibilă prin funcția thing) ci doar o valoare „indirectă" (în speță o listă), accesibilă doar prin anumite instrucțiuni. Ca dovadă, în linia (40) am creat o variabilă care poartă același nume (patrat), dar care nu se confundă cu lista de proprietăți. În linia (41) ne convingem că atât variabila cât și lista de proprietăți există și pot fi folosite simultan. Funcția plist generează, pe baza listei de proprietăți al cărei nume îl primește ca argument, o listă „normală" care are pe pozițiile impare numele proprietăților iar pe cele impare valorile acestora. Funcția sentence (pe scurt se) „concatenează" listele pe care le primește ca argumente.

Este deja destul de clar, cred, că listele de proprietăți sunt un fel de structuri de date libere, care colectează diverse informații de interes, numite proprietăți. O proprietate este un cuplu format dintr-un nume și o valoare. Extragerea valorii unei proprietăți (identificate prin nume) se poate face prin funcția gprop, ca în linia (38). Există și posibilitatea de a utiliza funcția plist pentru a genera o listă pe care o putem prelucra după voie, dar nu acesta este scopul.

Este foarte important să remarcăm că tentativa de a extrage valoarea unei proprietăți inexistente nu este penalizată cu un mesaj de eroare (vezi linia 39). Este foarte normal, deoarece în caz contrar aceste liste și-ar pierde puterea, care constă tocmai în libertatea de a folosi orice proprietăți dorim, fără să fim obligați să le declarăm în prealabil. Atenție: dacă proprietatea cerută nu există, Logo returnează implicit lista vidă, care însă ar putea fi valoarea unei proprietăți. Nu avem cum să discernem dacă proprietatea nu există (poate că a fost ștearsă, prin comanda remprop) sau dacă există și are ca valoare lista vidă.

Se pot imagina multe utilizări interesante pentru listele de proprietăți. Într-o aplicație care lucrează cu forme geometrice, listele de proprietăți ar putea fi utilizate pentru a memora valorile implicite pentru diferite forme, astfel încât diferitele instanțe ale formelor să nu poarte cu ele decât elementele specifice, economisind astfel spațiu și sporind portabilitatea aplicației. Pornind de la conversația precedentă, putem continua astfel:

```
(42)  ? pprop "forma "line 1
(43)  ? pprop "forma "fill "false
(44)  ? pprop "cerc "raza 100
(45)  ? remprop "patrat "poz
(46)  ? pprop "forma "poz [0 0]
(47)  ? remprop "patrat "col
(48)  ? pprop "forma "col :negru
```

În felul acesta am păstrat în lista de proprietăți patrat doar caracteristicile particulare ale pătratului (lungimea laturii), transferând celelalte proprietăți într-o listă de proprietăți mai generală, pe care am numit-o forma. Același lucru este valabil și pentru cerc.

Putem scrie proceduri prin care să generăm anumite variabile oarecum „tipizate":

```
(1) to patrat :nume :lista
(2) make word " :in :lista
(3) pprop word " :in "forma "patrat
(4) end
```

Singura „subtilitate" din această procedură constă în faptul că, deoarece make cere ca prim argument un literal, a trebuit să-l construim folosind funcția word pentru a plasa în fața numelei ghilimelele de rigoare. Apoi am creat o listă de proprietăți cu același nume, prin care vreau să specific că variabila nou creată este un pătrat.

Procedura pentru cercuri este definită la fel:

```
(1) to cerc :nume :lista
(2) make word " :in :lista
(3) pprop word " :in "forma "cerc
(4) end
```

Acum putem crea variabile care sunt de tipul „patrat" sau „cerc". Valorile lor trebuie să fie liste prin care să se precizeze proprietățile care diferă de cele implicite. De pildă:

```
? patrat "alfa [col [256 000 000] fill true]
? cerc "beta [poz [100 100]]
```

Iată în continuare o procedură care se folosește de aceste proprietăți:

```
(1) to deseneaza :in
(2) local "p make "p pen
(3) local "nume make "nume word " :in
(4) local "lista make "lista thing :nume
(5) ifelse memberp "poz :lista ~
(6)     [setpos first bf member "poz :lista] ~
(7)     [setpos gprop "forma "poz]
(8) ifelse memberp "col :lista ~
(9)     [setpencolor first bf member "col :lista] ~
(10)    [setpencolor gprop "forma "col]
(11) ifelse memberp "lin :lista ~
(12)    [setpensize first bf member "lin :lista] ~
(13)    [setpensize gprop "forma "lin]
(14) if equalp "patrat gprop :nume "forma ~
(15)    [ifelse memberp "lat :lista ~
(16)        [des_patrat first bf member "lat :lista] ~
(17)        [des_patrat gprop "patrat "lat]]
(18) if equalp "cerc gprop :nume "forma ~
(19)    [ifelse memberp "raza :lista ~
(20)        [des_cerc first bf member "raza :lista] ~
(21)        [des_cerc gprop "cerc "raza]]
(22) setpen :p
(23) end
```

Funcția member caută cuvântul care i se furnizează ca prim argument în lista pe care o primește ca al doilea argument. Dacă îl găsește, returnează partea din listă care

începe cu elementul căutat, altfel returnează lista vidă. Așadar, secvența

```
first bf member "ceva :lista
```

nu face altceva decât să returneze din lista dată elementul imediat următor celui egal cu „ceva". Este ușor de remarcat că, de fapt, imităm principiul de funcționare al primitivei gprop, folosind însă doar liste obișnuite.

Procedura `deseneaza` nu este spectaculoasă: verifică dacă proprietățile se găsesc în lista (simplă) desemnată de variabila care poartă numele pătratului sau cercului primit ca argument. Dacă da, folosește valorile de acolo, altfel le ia din lista de proprietăți corespunzătoare formei variabilei primite ca argument.

Ce face cu valorile corespunzătoare acestor proprietăți? Aici începe...

Grafica țestoasei

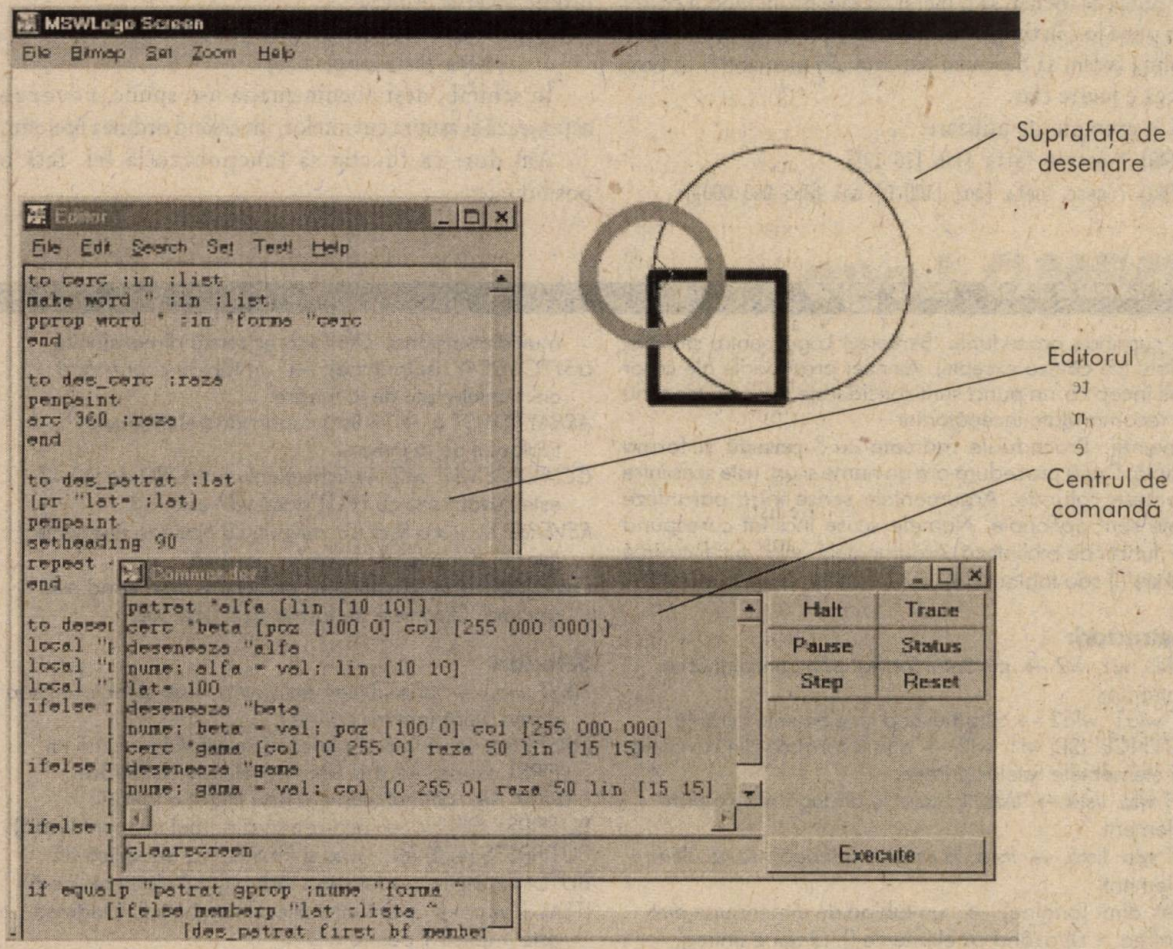
De ce „țestoasei"? Pentru că „dispozitivul" cu care se desenează în Logo se cheamă *turtle* (broască țestoasă). Desenul fiind poarta principală în Logo pentru copii, inventatorii limbajului au ales o personificare cât mai atractivă, cât mai simpatică („haioasă", dacă preferați). În unele implementări (Superlogo, de exemplu), pe ecran chiar este desenată o țestoasă. MSWLogo este mai sobru: țestoasa e reprezentată printr-un triunghi.

Ce face țestoasa? Duce cu ea o peniță și ascultă de comenzi simple, de genul: du-te înainte 100 de pași (`forward 100`), întoarce-te cu 30 de grade spre dreapta (`right 30`) sau du-te la poziția dată prin coordonatele 200 și -50 (`setxy 200 -50` sau `setpos [200 -50]`). Dacă penița este jos, în urma broaștei rămâne o linie. Țestoasa știe să manevreze penița pe baza unui alt set de comenzi simple: las-o jos (`pendown`), ridic-o (`penup`), pune-i cerneală galbenă (`serpencolor [255 255 0]`) și așa mai departe.

De fapt, modul de desenare este asemănător cu al unui plotter. Desigur, țestoasa știe să facă și lucruri pe care un plotter nu le poate face. De exemplu, să schimbe penița cu o radieră (`penerase`), după care, pe unde trece, șterge.

Din păcate, partea grafică are de-a face cu dispozitivul de afișare al calculatorului și, ca urmare, apar foarte multe elemente care sunt dependente de hardware, tratate în mod diferit de diferite implementări, ceea ce reduce foarte mult portabilitatea programelor. Totuși, principiile sunt aceleași, așa că zece minute în compania cărțuliei numite *User Guide* rezolvă toate problemele.

Toate exemplele din acest articol funcționează cu MSWLogo și există șanse minime să funcționeze sub un alt dialect (nici măcar în Berkeley Logo!). În această imple-



MSWLogo, un mediu de lucru freeware realizat de Softronics, Inc.

mentare, „pașii" țestoasei sunt de fapt pixeli, iar culorile se codifică în model RGB, adică o culoare se reprezintă ca o listă cu trei elemente având valori de la 0 la 255. Primul element exprimă „concentrația" de roșu, al doilea se referă la verde, iar al treilea la albastru. Forma peniței este, în această implementare, dreptunghiulară, deci grosimea liniei se exprimă printr-o pereche de numere. Astfel se explică argumentele procedurilor `setpencolor` în liniile 9-10 și `setpensize` în liniile 12-13. Observați că, de fapt, procedura `desenează`. nu desenează de fapt nimic, ci doar „pregătește terenul". Pentru desenarea efectivă sunt apelate două proceduri, `des_patrat` și `des_cerc`, care trebuie definite. A venit momentul s-o facem:

```
(1) to des_patrat :lat
(2) penpaint
(3) setheading 90
(4) repeat 4 ~
(5)     [forward :lat right 90]
(6) end
(1) to des_cerc :raza
(2) penpaint
(3) arc 360 :raza
(4) end
```

Sunt puține de zis. `Penpaint` spune țestoasei să ia penița (s-ar putea ca tocmai să fi lucrat cu radiera, în modul *erase*) și s-o pună jos, în timp ce `setheading` stabilește o orientare absolută (vrem să desenăm pătratul „în picioare"). În rest, cred că e foarte clar.

Un exemplu de utilizare:

```
(49) ? patrat "alfa [lin [10 10]]
(50) ? cerc "beta [poz [100 0] col [255 000 000]]
(51) ? deseneaza "alfa
(52) ? deseneaza "beta
(53) ? cerc "gama [col [0 255 0] raza 50 lin [15 15]]
(54) ? deseneaza "gama
```

Dacă secvența de mai sus o executăm folosind mediul MSWLogo, va rezulta desenul care se vede în captura de ecran din pagina precedentă (dacă facem abstracție de culori, din păcate). Desenul este de tip rastru (*bitmap*) și poate fi salvat ca fișier BMP.

A desena pătrate și cercuri nu-i o treabă grea nici în alte limbaje. Grafica țestoasei însă abia de-aici începe. Dacă țineți seama că în mai multe medii Logo (printre care și MSWLogo) puteți lucra cu mai multe țestoase deodată, că puteți scrie text, că puteți folosi imagini bitmap externe, că puteți lucra direct în 3D... devine mai atractiv, nu?

Iterativ versus recursiv

Și acum înapoi la lucruri serioase, adică la programare. Logo permite atât folosirea tehnicilor iterative, cât și a celor recursive. Ar fi interesant de făcut o comparație între ele.

Pentru aceasta, vă propun să încercăm să scriem o variantă a procedurii `reverse` (este o funcție de bibliotecă). Documentația ne spune că funcția primește la intrare o listă și că returnează o listă având aceleași elemente, dar în ordine inversă. Adică

```
(55) show reverse [alfa [beta gama] delta]
     [delta [beta gama] alfa]
```

În schimb, deși documentația n-o spune, `reverse` acționează și asupra cuvintelor, inversând ordinea literelor.

Am dori ca funcția să funcționeze la fel. Iată o posibilitate:

MEMO Logo

Lista cuprinde procedurile (Berkeley) Logo pentru structuri de date (cu câteva excepții). Atenție: procedurile ale căror nume încep cu un punct sunt considerate *periculoase* și nu sunt recomandate începătorilor.

Convenții: Procedurile marcate cu * posedă și forma multiplă. Dacă procedura are un nume scurt, este scris între paranteze rotunde. Argumentele scrise între paranteze drepte sunt opționale. Numele scrise înclinat corespund unor funcții de bibliotecă. Argumentele „wla" pot fi cuvinte (w), liste (l) sau tablouri (a).

Constructori:

WORD* w1 w2 → cuvântul format prin concatenarea intrărilor.
LIST* wla1 wla2 → o listă având ca elemente intrările.
SENTENCE* (SE) w1 w2 → o listă formată din cuvintele și elementele listelor primite.
FPUT wla lista → *lista*, la care se adaugă *wla* ca prim element.
LPUT wla lista → *lista*, la care se adaugă *wla* ca ultim element.
ARRAY dim [origine] → un tablou de dimensiune *dim* (întreg > 0) având ca elemente []. *Origine* (întreg, implicit 1) specifică indexul primului element.
MDARRAY dimlist [origine] → un tablou multidimensional. *Dimlist* precizează dimensiunile.
LISTTOARRAY list [origine] → un tablou cuprinzând elementele liste de la intrare.
ARRAYTOLIST a → o listă cuprinzând elementele tabloului de la intrare.
COMBINE w1 w2 → Echivalent cu WORD dacă *w2* este cuvânt sau cu FPUT dacă *w2* este listă.
REVERSE lw → o listă (un cuvânt) cu aceleași elemente (litere), dar așezate în ordine inversă.
GENSYM → un cuvânt unic de fiecare dată când este apelată. Cuvintele au forma „g1, „g2, etc.

Selectori

FIRST wla → primul element (pentru liste și tablouri) sau prima literă (pentru cuvinte).
FIRSTS list → o listă cuprinzând rezultatul aplicării lui FIRST asupra fiecărui element al listei de intrare.
LAST lw → ultimul element (sau literă) a intrării.
BUTFIRST (BF) lw → intrarea fără primul element (literă).
BUTFIRSTS (BFS) list → ca și FIRSTS, dar se aplică BF.
BUTLAST (BL) lw → intrarea fără ultimul element (literă).
ITEM i wla → elementul (litera) *i* al intrării. Originea este implicit 1 pentru *w* și *l*.
MDITEM ilist array → elementul tabloului MD selectat prin *ilist*.

```
(1) to invers :in
(2) make "rez []
(3) if wordp :in [make "rez "]
(4) make "lim count :in
(5) for [i 1 :lim 1] ~
(6)     [make "rez combine item :i :in :rez]
(7) op :rez
(8) end
```

Ideea este să începem cu o listă vidă (linia 2) sau cu un cuvânt vid, dacă argumentul este un cuvânt (linia 3), astfel încât să putem folosi procedura `combine`, care acționează ca `word` dacă al doilea argument sau ca `fput` dacă este listă). Comanda `for` este cea care parcurge pas cu pas intrarea. Primul argument este o listă cu elemente de control: primul este numele contorului (o variabilă locală pe care `for` o creează), urmând apoi limita inferioară, limita superioară și pasul. Al doilea argument este o listă de instrucțiuni care se execută pentru fiecare valoare a contorului.

Să încercăm acum și o variantă recursivă:

```
(1) to invers1 :in
(2) if emptyp :in [op :in]
(3) ifelse wordp :in ~
(4)     [op word invers1 bf :in first :in] ~
(5)     [op lput first :in invers1 bf :in]
(6) end
```

Nu am putut evita tratarea diferențiată a cazurilor în funcție de argumentul primit (așa cum face procedura `reverse` din biblioteca Logo), însă funcționează corect.

Varianta recursivă pare mai simplă, dar s-ar putea să fie mai greu de urmărit pentru cei care nu au lucrat suficient cu astfel de funcții. Argumentul care pledează însă pentru recursivitate este faptul că aceasta se pretează cel mai bine la procesarea listelor. Ce ar fi dacă am vrea ca funcția să inverseze și listele care sunt elemente ale listei de intrare?

```
(1) to invers2 :in
(2) if emptyp :in [op :in]
(3) ifelse wordp :in ~
(4)     [op word invers2 bf :in first :in] ~
(5)     [if wordp first :in ~
(6)         [op lput first :in invers2 bf :in] ~
(7)         [op lput invers2 first :in invers2 bf :in]]
(8) end
```

Câteva răspunsuri

Am început acest articol punând, la modul retoric, câteva întrebări. Două au rămas încă fără un răspuns explicit.

Ce se poate face cu Logo? Teoretic, orice. Practic, singura limitare este viteza (ca în cazul oricărui limbaj interpretat). MSWLogo dispune de posibilitatea de a crea în Logo interfețe Windows, de a programa elemente multimedia, chiar și comunicații TCP/IP. Printre exemplele care se livrează odată cu programul (totul încape pe o dischetă!) se află chiar și un compilator de Pascal!

De ce e bine să știți Logo? În primul rând ca să-l folosiți. Merită. Dar mai ales pentru că vă propune o altă viziune asupra programării. Veți programa mai bine în C, Pascal sau Java, pentru că aveți încă o sursă de inspirație.

❑

Mircea Sârbu este directorul editorial al editurii Computer Press Agora. Poate fi contactat prin e-mail la adresa msarbu@agora.ro.

PICK lw → un element (literă) aleatoriu din intrare.
REMOVE x lw → intrarea *lw*, din care s-au îndepărtat elementele egale cu *x*.
REMDUP lw → intrarea, din care s-au îndepărtat dublurile (se păstrează elementul din dreapta).

Modificatori

SETITEM i a x - Atribuie valoarea *x* elementului *i* al tabloului *a*.
MDSETITEM ilist a x - Ca și SETITEM, dar funcționează pe tablouri multidimensionale.
.SETFIRST list x - Înlocuiește primul element al listei *list* cu *x*. Nu verifică circularitatea.
.SETBF list x - Înlocuiește restul listei *list* (fără primul element) cu *x*. Nu verifică circularitatea.
.SETITEM i a v - Ca și SETITEM, dar nu verifică circularitatea.
PUSH numestiva x - Adaugă elementul *x* la stiva numită *numestiva* (stiva este o listă).
POP numestiva → cel mai recent element introdus (cu PUSH) și îl îndepărtează din stivă.
QUEUE numecoada x - Adaugă elementul *x* la coada numită *numecoada* (coada este o listă).
DEQUEUE numecoada → cel mai vechi element introdus (cu QUEUE) și îl îndepărtează din coadă.

Predicate

WORDP x → *true* dacă x este un cuvânt, altfel *false*.
LISTP x → *true* dacă x este o listă, altfel *false*.
ARRAYP x → *true* dacă x este un tablou, altfel *false*.
NUMBERP x → *true* dacă x este un număr, altfel *false*.
EMPTYP x → *true* dacă x este cuvântul vid (") sau lista vidă ([]), altfel *false*.
EQUALP x y → *true* dacă *x* și *y* au aceeași valoare.
BEFOREP w1 w2 → *true* dacă cuvântul *w1* este înaintea lui *w2* în ordine lexicografică.
.EQ x y → *true* dacă *x* și *y* desemnează același obiect.
MEMBERP x law → Dacă *law* este listă sau tablou returnează *true* când *x* este un element. Dacă *law* este cuvânt returnează *true* dacă *x* este subșir al acestuia.

Interogări

COUNT wla → numărul de elemente (sau litere) ale intrării.
ASCII char → codul ASCII al caracterului de la intrare.
CHAR int → caracterul care are codul ASCII de la intrare.
MEMBER x lw → porțiunea din *lw* începând de la prima instanță a lui *x* până la sfârșit.
LOWERCASE w → cuvântul *w* scris doar cu majuscule.
UPPERCASE w → cuvântul *w* scris doar cu litere mici.

The NEVER ENDING Story...

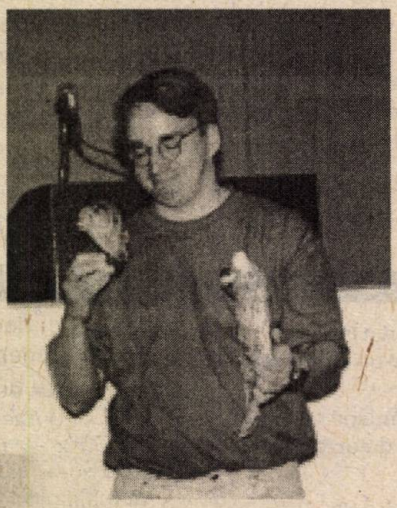
de Gabriel Proșcanu

Industria informatică este într-o extraordinară expansiune. Complexitatea sistemelor hardware și software crește într-un ritm amețitor iar resursele umane și financiare puse în joc depășesc puterile unei imaginații mai puțin exersate. Mai există oare șansa aventurii? Linus Torvalds, un student finlandez abia ieșit din adolescență, a dezlănțuit un joc care a cuprins întreaga lume.

De fiecare dată când auzim de o aplicație puțin comună sau de un server care deservește mii de clienți, prima întrebare care ne apare în minte este: „ce sistem de operare o avea calculatorul acela"? De obicei, răspunsul este: Unix sau Windows NT. Două sisteme diferite, cu două istorii diferite... În vreme ce NT nu are nici zece ani de existență, Unix domină lumea supercomputerelor de peste trei decenii.

Unix ? Care Unix ?...

Pe vremuri, marile companii nu cooperau atât de bine ca astăzi (vezi, de exemplu, Microsoft și Sun); de aceea, fiecare își crease propria variantă de Unix, incompatibilă cu a celuilalt. După vreo zece ani de încurcături, oamenii au început să se gândească la o oarecare standardizare și, cu puțină bătaie de cap, aceasta s-a realizat. Era, însă, așa cum am spus *o oarecare standardizare* și nu o standardizare pur și simplu. De aceea, în ziua de astăzi, există, în continuare, variante (*arome*, în jargonul cunoscătorilor) de Unix produse de Sun, de SCO, de IBM, de Hewlett-Packard, de Xerox și de alți giganți ai industriei software. De aceea, în clipa în care cineva se decidea să intre în lumea Unix, se vedea dependent de produsele unei anumite companii, ba, uneori, chiar și de un hardware specific. Și, pentru ca totul să fie cît mai atractiv, prețul unui sistem minim pornea de la 5.000 de dolari (doar sistemul de operare standard !)



Incredibilul se produce

În vara lui 1991, la câteva luni după ce își cumpărase primul său PC, un student finlandez a avut nevoie să transfere niște fișiere prin rețea. Computerul său era un 386, iar MS-DOS nu era tocmai mediul potrivit pentru asemenea operații. Studentul, pe numele său Linus Torvalds, și-a luat inima în dinți și a început să refacă lumea. *„Trebuia să scriu un driver de disc. L-am scris. Apoi, a trebuit să concep un sistem de fișiere, pentru a putea citi și scrie fișiere pe sistemul central Minix[1]. Bineînțeles, toate operațiile trebuiau să se desfășoare simultan, deci aveam nevoie de multitasking. Dar ce altceva decât Unix este un sistem de operare multitasking, cu sistem propriu de fișiere și cu drivere pentru periferice?"*

În acest fel s-a născut Linux - sau, cel puțin, kernel-ul său.

Ce e Linux ?

Linux este un sistem de operare *Unix-like*[2] pe 32 biți, care poate rula pe orice PC dotat cu un procesor Intel 386 sau mai bun. Oferă multitasking, lucru cu memoria virtuală, biblioteci și executabile partajate, alocare dinamică a memoriei, disponibilități excelente pentru lucrul în rețea și multe altele.

Aceleași facilități le oferă si alte variante de Unix. Linux se deosebește de acestea, prin câteva particularități uimitoare:

- în timp ce *aromele* respective costă multe mii de dolari, Linux este **gratuit**;
- Linux a fost, inițial, produsul unei singure persoane - Linus Torvalds - dar, pe măsură ce Linux se făcea tot mai cunoscut, mii de programatori din lumea întreagă (mult mai mulți decât angajații oricărei companii de soft din lume) au scris aplicații, au depanat sistemul de operare și au contribuit la crearea versiunilor ulterioare. Aici, un rol crucial l-a avut Internetul; comunicarea permanentă și rapidă a permis

cooperarea între entuziaștii care s-au dedicat proiectului inițiat de Linus Torvalds. Fără Internet, miracolul nu ar fi fost posibil.

Cum adică GRATUIT ?

Linux se conformează *licenței publice GNU*, adică poate fi copiat, modificat și distribuit liber. Nimeni nu poate impune restricții asupra distribuției, iar sursele programelor trebuie să fie mereu disponibile oricui dorește să le compileze sau să le modifice.

Dacă nu-l luați direct de pe Net (calea cea mai rapidă pentru cei care au acces la o conexiune T1 sau ceva similar), atunci vi-l puteți procura pe CD-ROM-uri. Sunt câteva companii care oferă atât sursele și executabilele Linux, cât și sute de aplicații; în acest caz, va trebui să plătiți suportul magnetic pe care vă sunt oferite programele (cel mult 50 de dolari). În jargonul Linux, acestea se numesc *distribuții*: RedHat, Slackware, Debian și diferă între ele doar prin arhivele oferite și prin programele de instalare.

Un sistem deschis

Linux nu rulează doar pe PC-uri X86. Mii de programatori lucrează neîncetat la portarea acestuia pe cele mai diverse arhitecturi, iar rezultatele sunt notabile. Nici un alt sistem de operare nu poate funcționa pe computere cu procesoare Power-PC, Motorola, DEC-Alpha sau MIPS. În plus, datorită numărului foarte mare de sisteme de fișiere recunoscute și a unui *boot-manager* inteligent, Linux poate conviețui cu alte sisteme de operare, în același computer.

❑

Note:

1) Minix este o **aromă** de Unix, creată de *Andrew Tannenbaum*, profesor de Știința Computerelor în Olanda și o celebritate în materie de sisteme de operare și de rețele.

2) De ce **Unix-like**? Deoarece „Unix" este proprietatea firmei X-Open și nu poate fi folosit de altcineva în scopuri comerciale.

Gabriel Proșcanu este redactor permanent al revistelor PC Report și Byte Romania. E-mail: gproscanu@agora.ro.

Linux: istoria

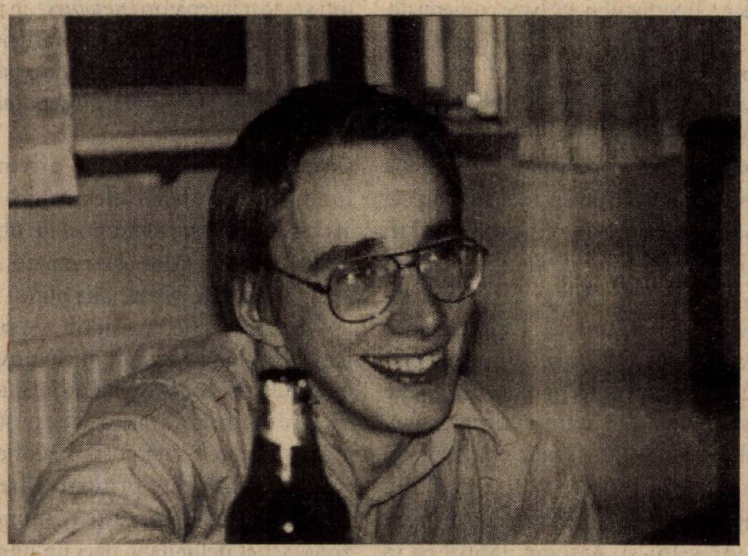
♦ În ianuarie 1991, Linus Torvalds își cumpără un 386. În aprilie, scrie o aplicație care realizează *task-switching* (comutarea între aplicații) Oricât ar părea de ciudat astăzi, acel proto-Linux nu făcea altceva decât să comute între două aplicații care scriau pe ecran - una „AAAA...", iar cealaltă „BBB...".

♦ Pe 3 iulie 1991, Linus postează un anunț pe comp.os.minix, cerând lămuriri despre felul în care ar putea afla ultimele reglementări POSIX (acea *oarecare standardizare*). În acest moment, Linus începuse să se gândească la aplicațiile la nivel de utilizator. În luna septembrie, sursele versiunii 0.01 au fost puse în `/pub/OS/Linux` de pe serverul `nic.funet.fi`. Deși această versiune nu putea fi rulată decât sub Minix și doar pe o mașină foarte asemănătoare cu cea pe care o avea Linus, a fost un bun semnal pentru ceilalți, care s-au simțit încurajați de faptul că le era pusă la dispoziție întreaga sursă a codului.

♦ Pe 5 octombrie a apărut versiunea 0.02, care includea câteva executabile, dar care era încă dependentă de Minix pentru a fi compilată. Nu era prevăzut lucrul cu dischete și nici cu memoria virtuală. În scurt timp a apărut versiunea 0.03, care corecta câteva probleme ale versiunii precedente.

♦ Pe 19 decembrie era gata versiunea 0.11, prima versiune care nu mai depindea de Minix. Doar discurile IDE erau suportate, dar se lucra intens și la SCSI. Acum apărea și un copyright (nu GNU, dar similar în multe privințe), care prevedea că este interzisă distribuirea contra cost, iar `tsx-11.mit.edu` devenea *mirror* pentru Linux.

♦ În martie 1992 apărea versiunea 0.95, care marca o evoluție importantă în dezvoltarea sistemului. Erau suportate tot felul de periferice și apărea X-Windows. Drumul către versiunile de astăzi (2.3.0...) se deschidea, plin de promisiuni, în fața programatorilor, tot mai numeroși, care începuseră să se intereseze de copilul minune al informaticii actuale...



E-MAIL: mesaje în ETER

de *Mircea Sârbu*

Adolescenții americani nu mai schimbă numere de telefon ci adrese de e-mail. Față de viteza și precizia poștei electronice, orice altă formă de corespondență va deveni curând desuetă. Chiar dacă astăzi în România accesul elevilor și studenților la e-mail este limitat (uneori din obtuzitate), cu siguranță mâine va fi altfel. Pentru că nu mai există cale de întoarcere.

De-a lungul evoluției sale, Internetul a trecut prin mai multe etape. La început, a fost o rețea militară, creată de US DoD (*Department of Defence* — Ministerul Apărării Statelor Unite). Ideea de la care s-a pornit a fost aceea de a introduce un grad ridicat de redundanță, astfel încât distrugerea unei părți a rețelei să nu afecteze funcționarea restului rețelei. Perioada următoare a aparținut comunității științifice și academice. Oamenii de știință au folosit rețeaua pentru a comunica între ei, pentru a distribui informații, pentru a face cunoscute rezultatele cercetării. Etapa actuală este cea a utilizării publice a Internetului, pe baze comerciale.

Dintre toate serviciile Internetului, poșta (sau mesageria) electronică (*e-mail* în engleză, dar din ce în ce mai mult și în română...) este cea mai simplă și cea mai răspândită. Trebuie spus aici că poșta electronică nu este apanajul exclusiv al Internetului: există rețele — locale sau de arie largă — private care dispun de sisteme de mesagerie electronică. De asemenea, majoritatea BBS-urilor permit utilizatorilor proprii să corespondeze între ei. Din perspectiva utilizatorului, folosirea acestor sisteme este similară cu utilizarea sistemului e-mail din Internet, exceptând modul de formare a adreselor și, desigur, faptul că în aceste sisteme nu se poate comunica decât în cadrul organizației respective. Dar expansiunea Internetului este foarte rapidă și majoritatea acestor sisteme sunt deja conectate la Rețeaua Globală.

Abecedarul e-mail

Cu siguranță mulți dintre cititori folosesc cu dezinvoltură poșta electronică. Există însă mulți care încă nu au avut posibilitatea, interesul sau poate curajul să-i cunoască avantajele. Cert este că, în scurt timp, e-mail-ul va deveni un instrument indispensabil pentru oricine, așa că o introducere rapidă este, cred eu, necesară. Desigur, toți am fost începători (*newbies* - în jargonul Rețelei) și câteva sfaturi la timpul potrivit ne-ar fi scutit de o parte dintre greșelile inerente ale începutului.

Conectarea. În multe privințe, funcționarea poștei electronice se aseamănă cu serviciile poștale obișnuite. Analogia este extrem de expresivă, așa că o voi detalia puțin.

În general, fiecare dintre noi suntem clienții unui anumit oficiu poștal, de regulă cel mai apropiat de locuință. Fiecare dintre noi avem o cutie poștală acasă și/sau o căsuță poștală la oficiul poștal, identificate printr-o adresă. Oficiul poștal deservește o anumită zonă, strict delimitată, deci un număr bine precizat de adrese.

În cazul poștei electronice, rolul oficiului poștal este jucat de un calculator (de regulă o mașină Unix sau Windows NT) care rulează un program numit (la modul generic) ***server de e-mail***. Suntem clienții acestui server dacă dispunem de un *cont* pe acest server. Aceasta implică faptul că avem o *adresă e-mail* prin care putem fi identificați și un spațiu de stocare unde sunt memorate mesajele (*mail-box* — cutie poștală).

Un oficiu poștal are două interfețe: una spre clienții său (corespunzători adreselor care-i sunt arondate) și una spre ansamblul sistemului poștal. Prima interfață implică distribuirea corespondenței sosite spre clienții oficiului și preluarea corespondenței acestora în vederea expedierii. A doua interfață o reprezintă relația directă sau indirectă (prin intermediul altor oficii poștale, oficii de triere etc.) cu celelalte oficii poștale.

În mod analog, serverul de e-mail dispune de două protocoale diferite, corespunzătoare acestor interfețe. Astfel, comunicarea cu clienții se face, în marea majoritate a cazurilor, prin protocolul POP3 (*Post Office Protocol 3*), care asigură distribuirea mesajelor (acesta este termenul prin care ne referim la o „scrisoare electronică")

recepționate către clienții proprii cărora le sunt destinate și preluarea mesajelor expediate de aceștia. Interfața cu alte servere de e-mail din Internet se face prin protocolul SMTP (*Simple Mail Transfer Protocol*), prin care serverul recepționează și expediază corespondența clienților săi în și din Rețeaua Globală.

De regulă, utilizatorul are de-a face cu un singur server de e-mail (altfel spus: serverul POP3 și serverul SMTP rulează pe aceeași mașină). Este însă posibil ca expedierea să se facă printr-un alt server SMTP. Analogia cu poșta obișnuită este evidentă: nu suntem obligați să trimitem scrisorile prin intermediul oficiului poștal căruia îi suntem clienți. Le putem expedia de oriunde, dar le putem primi doar acasă.

Transferul mesajelor. La modul general, atât poșta obișnuită cât și poșta electronică funcționează pe principiul „stochează și trimite mai departe" (*store and forward*). Pentru a ilustra acest principiu, vom analiza în paralel drumul unei scrisori și cel al unui mesaj.

Utilizatorul scrie de obicei scrisoarea acasă, o pune într-un plic pe care scrie adresa completă a destinatarului și o pune într-o cutie poștală. Din timp în timp, funcționarii poștei adună scrisorile și le depun la un oficiu poștal. În cazul unui mesaj e-mail, acesta este compus de regulă pe stația (calculatorul) utilizatorului, prin intermediul unui program numit client e-mail (sau *mailer*, cum i se mai spune). Antetul mesajului este echivalentul plicului și va conține informațiile publice necesare (vezi caseta „Anatomia unui mesaj"). Inițial mesajul este stocat local, dar la prima conectare la serverul SMTP, acesta este preluat și stocat pe server. Dacă utilizatorul este conectat permanent cu serverul (de regulă, într-o rețea locală) expedierea pe server se poate face și imediat, dar dacă legătura se face prin linie telefonică comutată (prin intermediul unui modem) mesajele vor fi preluate de server la prima conectare.

La oficiul poștal se face o triere a scrisorilor după adrese. Scrisorile loco sunt trimise direct oficiului poștal care deservește adresa expeditorului. Celelalte scrisori sunt trimise unor oficii poștale care deservesc zonele teritoriale corespunzătoare adreselor. Există o ierarhie: toate scrisorile destinate adreselor dintr-o zonă (de pildă un județ) sunt trimise la un anumit oficiu, care le adună, le triază și le trimite mai departe, spre oficii poștale, până la oficiul poștal care deservește direct destinatarul scrisorii.

În cazul unui mesaj e-mail, serverul procedează cam la fel. Dacă destinatarul este un client al său, mesajul este plasat direct în mailbox-ul acestuia. Celelalte mesaje sunt trimise unor niveluri Internet superioare, unde sunt triate în funcție de adrese și redistribuite, ajungând adesea să treacă prin multe servere intermediare, până când ajung la serverul de e-mail al destinatarului.

În fine, ultimul pas în drumul scrisorii este cel de la oficiul poștal la destinatar. Dacă adresa menționează o căsuță poștală, atunci scrisoarea este depusă acolo, altfel poștașul o duce și o pune în cutia poștală a destinatarului.

În termenii poștei electronice, acest ultim pas corespunde preluării mesajului de către programul client e-mail cu care lucrează destinatarul (această operație se face la prima conectare a mailer-ului la serverul POP3). Odată ajuns în calculatorul destinatarului, acesta poate să-l citească, să-l păstreze, să-l șteargă etc.

Servicii speciale. Mai există similitudini. Ce se întâmplă dacă, dintr-un motiv sau altul (de pildă, destinatarul și-a schimbat adresa) scrisoarea nu poate ajunge la destinație? În mod normal, ea este trimisă înapoi. Un mesaj poate la rândul lui să întâmpine anumite dificultăți. De exemplu, unul dintre serverele intermediare este indisponibil, caz în care se recurge la trasee ocolitoare. Dacă serverul desti-

Anatomia unui mesaj

Un mesaj este compus din două părți: antetul și corpul.
Antetul mesajului cuprinde la rândul său două părți:
(a) o parte publică (plicul) cuprinzând informații utile pentru transportul mesajului;
(b) o parte privată, cuprinzând informații utile pentru receptorul uman sau programul client de e-mail al acestuia.

Plicul:

To:	Adresele de e-mail ale receptorilor primari
Cc:	Adresele de e-mail ale receptorilor secundari
Bcc:	Adresele de e-mail pentru copii confidențiale
From:	Persoana care a creat mesajul
Sender:	Adresa de e-mail de la care a fost trimis mesajul
Recieved:	Linie adăugată de fiecare nod intermediar prin care a trecut mesajul
Return-Path:	Poate fi folosită pentru a specifica o cale de întoarcere la expeditor

Câmpuri private:

Date:	Data și ora la care a fost expediat mesajul
Reply-To:	Adresa de e-mail la care trebuie trimise răspunsurile
Message-Id:	Număr unic care identifică mesajul
In-Reply-To:	Identificatorul mesajului la care se răspunde prin acest mesaj
References:	Alți identificatori de mesaje relevanți
Keywords:	Cuvinte cheie alese de utilizator
Subject:	Scurt cuprins al mesajului

nație este indisponibil, expeditorul primește un mesaj de avertisment de la ultimul server intermediar, care însă va mai insista un timp (de obicei câteva zile) să intre în legătură cu serverul destinație. Dacă adresa este greșită (de pildă destinatarul nu mai are cont pe serverul menționat), expeditorul primește un mesaj prin care este înștiințat de acest lucru.

Există și posibilitatea de a trimite mesaje „recomandate", caz în care expeditorul primește un mesaj prin care i se confirmă în mod explicit livrarea mesajului către destinatar. Mai mult, unele programe pot solicita destinatarului să confirme citirea mesajului, dar funcționarea acestei facilități depinde de programul client folosit de destinatar.

Dincolo de similitudinile cu poșta obișnuită, mesageria electronică oferă o serie de servicii specifice, care nu au echivalent printre serviciile poștale obișnuite.

Cea mai simplă astfel de facilitate o reprezintă posibilitatea de a trimite același mesaj mai multor destinatari simultan. Această facilitate permite și crearea unor așa-numite „liste de discuții".

Alte facilități importante sunt atașarea fișierelor, redirectarea automată a mesajelor, filtrarea mesajelor în funcție de diferite criterii (combinată uneori cu răspunsuri automate), crearea de pseudonime (*alias*-uri) etc.

Funcțiile clienților de e-mail

Odată ce ne-am făcut o idee generală despre funcționarea poștei electronice în ansamblul ei, putem să ne concentrăm asupra programelor client de poștă electronică, cele cu care interacționăm în mod uzual.

Ca simpli utilizatori, este bine să știm că există multe astfel de programe, atât comerciale cât și *shareware* sau *freeware*. Vremea când ne conectam prin *telnet* la server și foloseam programe Unix (de gen *mail* sau *Pine*) a trecut. Astăzi se folosesc preponderent programe cu interfață grafică prietenoasă și intuitivă (Windows, Mac, Motif etc.), dar este bine să ținem seama de faptul că, de regulă, nu știm ce programe, ce sisteme de operare și ce calculatoare folosesc corespondenții noștri. În ciuda aparențelor, acesta nu este un dezavantaj, ci poate chiar „punctul forte" care a dus la expansiunea extraordinară a poștei electronice (și a Internetului în general): faptul că transcede platformele de calcul.

În linii mari, un client de e-mail trebuie să ne permită să comunicăm cu serverul, să vizualizăm mesajele pe care le primim, să compunem mesajele pe care le expediem și să ne organizăm într-un mod cât mai convenabil mesajele. Toate aceste funcții sunt însă strâns împletite.

Comunicarea. În principiu, această funcție se realizează automat. Trebuie totuși să știm că legătura cu serverul nu este permanentă, chiar dacă suntem conectați cu acesta printr-o rețea (eventual cuprinzând și linii telefonice închiriate). De cele mai multe ori, conectarea la server se face la anumite intervale de timp, ocazie cu care se transferă mesajele trimise și se aduc mesajele primite, după care programul eliberează legătura. Stabilirea acestui intervalul este lăsată la latitudinea utilizatorului (depinde de numărul mesajelor vehiculate și de calitatea rețelei).

Majoritatea programelor permit ca de fiecare dată când se expediază un mesaj să se stabilească legătura și mesajul să fie transmis imediat. Această opțiune este însă indicată doar în cazul în care conexiunea se face prin rețea locală.

Pentru utilizatorii care sunt conectați la server prin linie telefonică închiriată (marea majoritate a celor care utilizează calculatorul acasă și sunt conectați la Internet printr-un furnizor comercial), varianta optimă este conectarea explicită, pentru a minimiza utilizarea liniei telefonice (care costă, și nu puțin). Este de amintit că în acest caz, momentul „expedierii" mesajului nu coincide cu transmiterea lui efectivă (avem posibilitatea să ne răzgândim). Unele programe (de pildă Eudora) semnalează acest lucru prin schimbarea comenzii *Send* (trimite) cu *Queue* (pune în așteptare).

Vizualizarea. Este o funcție simplă, prin care putem să citim mesajele. Dacă mesajul este un simplu text ASCII pe 7 biți (cazul cel mai uzual) nu avem prea multe probleme. Este indicat să stabilim ca font de afișare (dacă programul permite acest lucru) un tip de caractere monospațiate (toate literele au lățime egală, de exemplu *Courier*) pentru că adesea cei care compun mesajele utilizând terminale alfanumerice recurg la alinieri ale textului pentru a-i spori lizibilitatea. Este bine să maximizați fereastra aplicației (sau măcar cea care afișează textul), pentru că adesea veți întâlni mesaje care conțin <CR> la capătul fiecărei linii, ceea ce conduce la o lectură cel puțin incomodă dacă liniile nu încap integral în fereastra de afișare.

Majoritatea *mailere*-lor actuale au posibilitatea să afișeze și text formatat (în principal HTML). Standardele însă nu sunt încă foarte stabile și se întâmplă adesea ca *viewer*-ul clientului nostru de e-mail să nu înțeleagă perfect formatarea realizată de *mailer*-ul corespondentului nostru. Și viceversa. Să sperăm că timpul va rezolva aceste probleme.

Singurul aspect mai complex al funcției de afișare este cel legat de fișierele binare atașate mesajelor. În cazul celor mai multe programe Windows95 sau Macintosh, lucrurile se desfășoară simplu: în mesaj vor apărea iconuri corespunzătoare tipurilor de fișiere atașate. De regulă, un dublu-clic pe icon lansează aplicația potrivită. Dacă binarul este un executabil, atunci îl lansează în execuție.

Compunerea. Această funcție este realizată cu ajutorul unui editor specializat, care cuprinde o parte de procesare de text și o parte de comenzi specifice. Unele programe (de pildă Microsoft Exchange) permit utilizarea unui alt procesor de text (cum ar fi Word for Windows) dar, în general, cel înglobat este suficient.

Cum procesarea de texte o cunoaș-

tem cu toții, să vedem elementele care sunt specifice clienților de e-mail. În primul rând, vom observa că editarea unui mesaj începe cu completarea unor câmpuri generice, care formează antetul mesajului (*header*). Acesta este echivalentul electronic al plicului: conține informațiile „publice", de care întregul sistem de mesagerie electronică are nevoie pentru a manevra scrisoarea. Iată în continuare cele mai importante dintre aceste câmpuri:

To: Aici va fi scrisă adresa destinatarului, care de obicei are forma *cutie_postală@locație* (din fericire, programele ne permit să exprimăm adresa și sub o formă mai inteligibilă: scriem în clar numele persoanei după care scriem adresa între paranteze unghiulare). Trebuie însă amintit faptul că anumite rețele, cum ar fi CompuServe, folosesc modalități diferite de adresare. Este important de subliniat că, spre deosebire de plicul obișnuit, aici putem scrie mai mulți destinatari, astfel încât mesajul va fi trimis către fiecare dintre ei.

Cc: Este prescurtarea de la *Carbon copy* - copie la indigo. Aici putem scrie adresele persoanelor care vrem să primească o copie a scrisorii. De obicei este vorba de persoane care sunt vizate de conținutul mesajului, care ar putea fi interesate de informațiile din mesaj, care monitorizează un proiect, sau oricine altcineva care considerăm că trebuie să afle conținutul mesajului. Persoanele ale căror adrese sunt trecute în câmpul *Cc:* se mai numesc „destinatarii secundari".

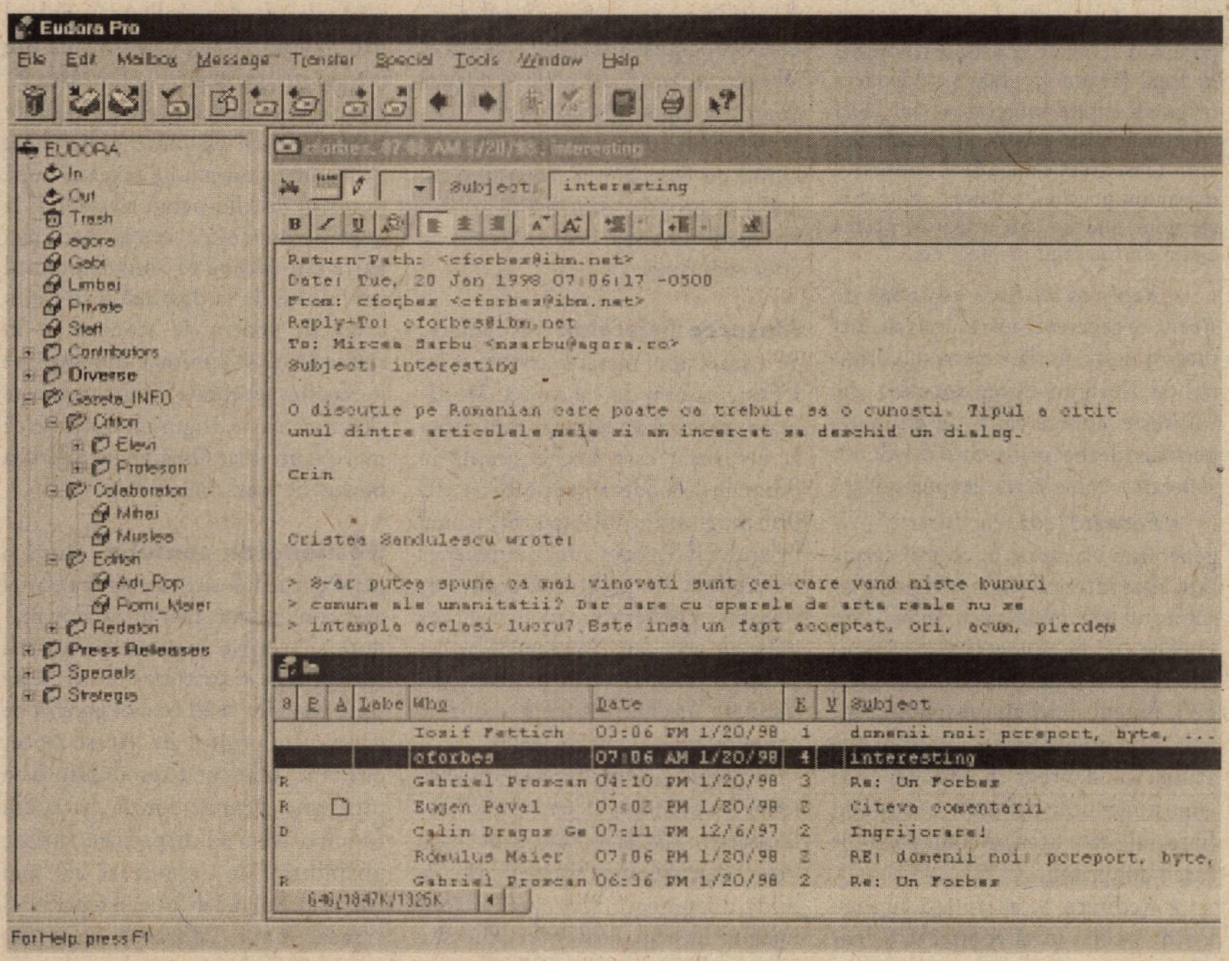
Bcc: Este prescurtarea de la *Blind carbon copy* (copie confidențală). Ca și în cazul precedent, câmpul poate cuprinde adresele unor destinatari secundari. Diferența față de copia simplă constă în faptul că destinatarii primari și secundari ai mesajului nu vor ști că mesajul este recepționat de acești corespondenți. Este bine să țineți seama de faptul că există posibilitatea ca un mesaj pe care l-ați primit să fi fost destinat și unor alte persoane decât cele pe care le vedeți în antetul mesajului.

Subject: Câteva cuvinte care să evoce subiectul mesajului.

După ce am completat aceste câmpuri, nu ne mai rămâne decât să editâm mesajul propriu-zis.

Să trecem în continuare în revistă și cele mai importante comenzi de care dispun programele de poștă electronică:

• *Reply* (răspuns, replică). Această



Un client de e-mail tipic (în imagine Eudora), permite vizualizarea sub formă de listă a mesajelor (fereastra din dreapta jos), vizualizarea și editarea unui mesaj (fereastra din dreapta sus), precum și organizarea întregii corespondențe într-o structură (în acest caz arborescentă) de „dosare" sau „foldere" (fereastra din stânga).

comandă este disponibilă când citim un mesaj sau când am selectat un mesaj dintr-unul dintre „dosarele" tematice (vezi mai jos), mesaj pe care-l vom numi în continuare „original". Rezultatul comenzii este că se lansează compunerea unui nou mesaj, completându-se însă automat anumite câmpuri ale antetului. În primul rând, câmpul *To:* va fi completat (de obicei) cu adresa expeditorului mesajului căruia îi răspundem. Dacă originalul conține o adresă în câmpul *Reply-To* din antet, atunci aceasta va fi adresa la care va fi trimis mesajul. În câmpul *Subject* va fi copiat subiectul mesajului original, dar se va adăuga în față „Re:". În fine, corpul mesajului va fi „citat"(*quote*) în întregime, adică va rămâne în editor, dar fiecare linie va fi prefixată de caracterul „>" (cazul cel mai comun, dar există și alte convenții pentru citare). În felul acesta putem să răspundem „punct-cu-punct", păstrând citatele relevante din mesajul original, putem să scriem o scrisoare compactă în față, lăsând la sfârșit scrisoarea originală citată integral (astfel încât corespondentul nostru să poată să-și amintească cât mai exact contextul mesajului pe care l-a trimis), putem să ștergem (parțial sau integral) partea citată din mesajul original etc.

• *Reply to all.* Spre deosebire de *Reply,* se generează un răspuns adresat tuturor destinatarilor mesajului original (și, desigur, expeditorului). Se folosește atunci când un grup de persoane dezbat prin e-mail un subiect și fiecare trebuie să afle ce spun ceilalți.

• *Forward* („dă mai departe"). Se generează un mesaj în corpul căruia este citat întregul mesaj original, iar subiectul este prefixat cu „FW:". Ne rămâne sarcina să completăm câmpurile rezervate destinatarilor (*to, cc, bcc*). Putem să editâm mesajul original, să-l comentăm etc. Se folosește atunci când vrem să aducem și la cunoștința altor persoane conținutul (integral sau parțial) al unui mesaj pe care l-am primit.

• *Redirect.* Se aseamănă cu *Forward.* Va trebui să completăm doar adresa destinatarilor. Spre deosebire de *Forward*, subiectul și corpul mesajului rămân aceleași (dar pot fi editate). Noii destinatari vor primi mesajul ca și cum le-ar fi fost trimis direct, cu singura diferență că la numele și adresa expeditorului se adaugă textul *„by way of"* urmat de numele și adresa celui care a făcut redirectarea (din motive evidente: acesta a avut posibilitatea să altereze conținutul mesajului original).

Ultimele observații legate de funcția de compoziție:

- multe dintre programele actuale dispun de numeroase funcții de formatare a textului (cel mai adesea HTML). Folosiți-le cu precauție: nu toate *mailer*-ele știu să interpreteze aceste formatări. Ați vrut să scrieți o scrisoare frumoasă și ați reușit să produceți una ilizibilă.

- Există posibilitatea să vă compuneți o „semnătură" (câteva linii de text stocate de obicei într-un fișier ASCII) care poate fi adăugată automat la sfârșitul fiecărui mesaj ce-l expediați. Unele programe îl adaugă de îndată ce începeți editarea, dar altele îl adaugă doar în momentul expedierii. Verificați modul de funcționare al programului cu care lucrați (trimiteți un mesaj pe propria adresă), altfel riscați să expediați mesaje nesemnate sau cu două semnături.

Atașarea fișierelor. Posibilitatea de a atașa unui mesaj fișiere de orice fel este extrem de valoroasă. Modalitatea practică de atașare a unui fișier la un mesaj este foarte simplă în programele moderne: se poate face fie prin *drag-and-drop* („tragem" iconul fișierului în fereastra de compunere a mesajului), fie printr-un dialog de selectare a fișierului (identic cu cel obișnuit prin care programele ne cer să specificăm fișierul pe care să-l încarce). Trebuie să ținem seama de anumite aspecte mai delicate.

Întregul sistem de e-mail din Internet a fost conceput pentru a vehicula caractere ASCII pe 7 biți, astfel încât atunci când trimitem fișiere binare (de pildă documente Word sau Excel, imagini bitmap, secvențe audio sau video, arhive comprimate, programe executabile etc.) trebuie să recurgem la o codificare a acestora care să le protejeze de eventuale alterări în timpul transferului. Există mai multe forme de codificare, dintre care cele mai utilizate sunt *uuencode* și *binhex.* Ambele transformă informația binară în coduri ASCII. La destinație, fișierul trebuie decodificat (majoritatea *mailer*-elor fac automat codificarea și decodificarea). Din fericire, în ultima vreme a căpătat o largă răspândire o nouă soluție, cunoscută sub numele MIME (*Multipurpose Internet Mail Extension*) care rezolvă automat toate aspectele legate de transmiterea datelor binare.

O ultimă observație: fișierele atașate pot să crească foarte mult dimensiunea mesajelor (teoretic nu există limită superioară). Este însă posibil ca pentru corespondentul nostru, recepționarea unui mesaj de mari dimensiuni să nu fie un motiv de bucurie. Dacă acesta este conectat printr-o linie telefonică de slabă calitate (cazul cel mai comun) s-ar putea să stea ore întregi pe line pentru a descărca fișierele, cu consecințele de rigoare asupra facturii telefonice (să nu uităm că Romtelecom nu ne taxează după numărul de kilo-octeți transferați ci după timpul de conectare). Pe de altă parte este posibil ca administratorul serverului de la destinație să fie în criză de spațiu de stocare și, în consecință, să limiteze dimensiune mesajelor acceptate. Concluzia este că e bine să ne asigurăm dinainte că nu vom produce nimănui necazuri cu mesaje de mari dimensiuni.

Gestionarea contului. După o perioadă de folosire intensivă a poștei electronice, numărul mesajelor primite și trimise (o copie a fiecărui mesaj trimis se păstrează local) crește și ne vine din ce în ce mai greu să ne descurcăm printre ele. Acest aspect este bine cunoscut de producătorii de programe client de e-mail, așa că din fericire avem la dispoziție variate posibilități de organizare cât mai eficientă. Soluția de bază o reprezintă crearea unor „dosare" sau „cataloage" (*folders* — uzual se folosește deja termenul „romglez" folder, foldere) în care putem plasa aceste mesaje. Conținutul unui folder este

vizualizat sub formă de tabel, cu un mesaj pe fiecare rând. Plasare unui mesaj într-un folder se face fie prin *drag-and-drop*, fie prin comenzi simple, specifice programului.

Cele mai multe programe dispun de câteva foldere predefinite, al căror nume variază de la un program la altul, chiar dacă funcționalitatea este mereu aceeași. Astfel, există întotdeauna un dosar unde sunt automat plasate noile mesaje sosite (numit de regulă *Inbox* sau *In*) și un dosar unde sunt plasate copii ale mesajelor trimise (numit adesea *Outbox* sau *Sent messages*). În fine, un alt dosar predefinit este cel în care se păstrează mesajele pe care le-am șters (numit *Deleted messages* sau *Trash* sau altfel). Este bine de știut că mesajele pe care le ștergem mai sunt disponibile până când le ștergem definitiv (cu o comandă de genul *Empty trash*).

În afară de folderele predefinite, avem posibilitatea de a ne crea propriile foldere, specifice activităților pe care le desfășurăm. Unele programe ne permit structurarea ierarhică a dosarelor, astfel încât acestea ajung să se asemene foarte mult cu structura de directoare a discurilor, cu diferența că în locul fișierelor aici vom găsi mesaje.

În cadrul unui folder, avem posibilitatea să sortăm mesajele după criterii diverse (după dată, după numele expeditorului sau al destinatarului, după subiect, după dimensiune etc.). Anumite marcaje ne permit să identificăm cu ușurință mesajele care nu au fost încă deschise, cele care conțin fișiere atașate, cele la care am răspuns, cele pe care le-am trimis mai departe, cele pe care le-am redirectat etc.

Facilități speciale. Am grupat în această categorie facilități care sunt specifice programelor, dar care nu-și au corespondent în manevrarea mesajelor în Internet. Altfel spus, aceste facilități au valoare locală.

• *Agenda cu adrese.* Majoritatea *mailer*-elor oferă facilitatea de a vă stoca adresele corespondenților, de cele mai multe ori împreună și cu alte informații (adresa poștală, numere de telefon și fax, observații etc.). Facilitățile de grupare a adreselor (tot un fel de foldere), de sortare și de căutare sunt comune. Unele programe oferă posibilitatea de a crea și adrese colective (sau *liste de distribuție*).

• *Filtre.* Posibilitatea de a filtra automat mesajele este adesea de mare valoare, mai ales atunci când volumul corespondenței este mare. Filtrarea se face pe baza unor condiții specificate de utilizator, cum ar fi prezența unui anumit șir de caractere într-unul din câmpurile antetului sau în corpul mesajului, precum și condițiile în care se aplică (de pildă mesajelor primite sau mesajelor trimise). Dacă condițiile sunt îndeplinite, atunci se execută anumite acțiuni specificate de utilizator (de exemplu se transferă mesajul într-un anumit folder, i se atribuie o anumită prioritate, este redirectat la o anumită adresă, se răspunde cu un text standard etc.).

• *Confidențialitate.* Toate programele de mesagerie electronică oferă utilizatorului posibilitatea (ba uneori

Cifrarea cu cheie publică

În sisteme de cifrare convenționale, cifrarea și descifrarea se bazează pe o cheie unică, aceeași atât la cel care cifrează mesajul, cât și la cel care îl descifrează. Inconvenientul major este că, într-un fel oarecare, destinatarului unui mesaj cifrat trebuie să îi parvină și cheia. Acest lucru neputând fi realizat pe căi nesigure - cum ar fi poșta electronică sau telefonul - înseamnă că există o cale de comunicare sigură, și atunci de ce ar mai fi nevoie de cifrare?...

Cifrarea cu cheie publică se bazează pe o pereche de chei de cifrare, una secretă și una publică. Proprietatea de bază a celor două chei este că ele sunt complementare: textul cifrat cu cheia secretă poate fi descifrat numai cu cheia publică și invers. În plus, cunoașterea uneia dintre chei nu este suficientă pentru a o putea deduce pe cealaltă. Modul de funcționare devine limpede: cheile publice sunt publice și pot fi transmise tuturor celor interesați, memorate asemenea unor numere de telefon etc. Oricine dorește să transmită un mesaj confidențial va cifra mesajul utilizând cheia publică a destinatarului. Nimeni în afară de deținătorul cheii secrete - adică destinatarul - nu va putea descifra mesajul, nici măcar cel care l-a cifrat...!

Schema poate fi folosită și pentru autentificare. Pentru a semna un document autentic, semnatarul îl va cifra folosind cheia sa secretă. Oricine va putea descifra mesajul utilizând cheia publică a autorului, având astfel certitudinea că textul este un text autentic, fără adăugiri sau ștersături. Textul descifrat poate fi desigur editat, dar în mod cert nu mai poate fi transmis mai departe ca având semnătura autorului original - pentru că pentru aceasta ar trebui cunoscută cheia secretă a autorului... Pe de altă parte, autorul nu se poate scuza că textul nu i-ar aparține: dacă poate fi descifrat folosind cheia lui publică, înseamnă că autorul este pe deplin responsabil pentru fiecare frază sau literă din textul respectiv.

Combinarea celor două procedee permite atât confidențialitate (destinatarul este singurul care poate citi mesajul) cât și autentificare (este cert că mesajul vine de la semnatar). Pentru aceasta, mesajul este cifrat în două faze: întâi, folosind cheia secretă a autorului, care „semnează" astfel mesajul, iar apoi utilizând cheia publică a destinatarului. La recepționarea mesajului, destinatarul îl va descifra utilizând întâi cheia sa secretă și apoi cheia publică a autorului.

(Iosif Fettich)

chiar îl obligă) să folosească o parolă pentru a-și proteja intimitatea corespondenței (în engleză: *privacy*). Ca regulă generală: este bine să alegeți parole care să nu poată fi ușor ghicite (de pildă, să nu folosiți variații ale numelui). Ca să n-o uitați, vă puteți nota în clar, într-un carnet, un element care să vă ajute să v-o amintiți (dacă ați ales o combinație bazată pe numele pisicii vecinilor, vă notați doar „pisică"). Nu încredințați nimănui parola. Trebuie însă spus din capul locului că, la ora actuală, nu există nici o garanție că nimeni nu are posibilitatea să ne citească mesajele, ba chiar să scrie unele în locul nostru. Cea mai simplă cale prin care ne putem apăra de intruziuni o reprezintă cifrarea mesajelor, ceea ce presupune însă o înțelegere prealabilă cu corespondenții (ceea ce nu este nici ușor și nici sigur).

O cale mai ușoară o reprezintă cifrarea cu chei publice și chei private. În acest scop puteți utiliza programul *freeware* numit PGP (*Pretty Good Privacy*), disponibil pe diverse situri Internet (vezi *„Cifrarea cu cheie publică"*).

De reținut: faptul că cineva își cifrează corespondența nu înseamnă că are ceva de ascuns. Dreptul la intimitate este recunoscut oricărui cetățean și fiecare trebuie să-l apere.

• *Diverse.* Imaginația proiectanților a dus la o adevărată inflație de facilități (multe dintre ele complet inutile) oferite de produsele software actuale. Mă voi limita doar la câteva, mai des folosite: posibilitatea de a eticheta mesajele pe baza unor categorii prestabilite, posibilitatea de a le atribui anumite priorități (ignorate la transferul de mesaje prin Internet dar poate relevante pentru destinatar, dacă *mailer*-ul său le recunoaște), posibilitatea de a programa printr-un limbaj propriu (de regulă *scripting*) anumite acțiuni, posibilitatea de a utiliza șabloane și „personalități multiple" (de exemplu Eudora Pro îmi permite să-mi configurez două sau mai multe stiluri de corespondență: una pentru mesajele oficiale, una pentru corespondența privată etc.).

Post restant

Una dintre inovațiile cele mai la modă în domeniul poștei electronice o reprezintă posibilitatea de a utiliza căsuțe poștale accesibile prin WWW. Programul client e-mail este în acest caz un applet Java, care poate fi rulat de orice browser Web care „știe" Java (*Java enabled*). Există situri (de exemplu Hotmail) care oferă oricui, gratuit, posibilitatea de a-și deschide o căsuță poștală, oferind în acest scop spațiu și servicii de administrare. Mai mult, din aceste căsuțe poștale se pot accesa și eventualele alte căsuțe poștale de care utilizatorul dispune, oriunde în lume. Avantajul constă în faptul că oriunde în lume avem posibilitatea să ne întreținem corespondență, cu singura condiție de a avea acces la WWW. În tot mai multe țări apar terminale Web cu acces public, iar cafenelele Internet sunt tot mai răspândite, chiar și în România.

Semnături

Câteva sugestii de semnături pentru mesajele voastre:

1. Propria mea semnatură „oficială". O consider suficient de „aerisită", oferind totodată și datele de contact necesare.

```
Mircea Sarbu
-------------------------------------------------------------
director editorial                    Computer Press Agora
                                      --------------------
                                      tel: +40-(0)65-166516
                                      fax: +40-(0)65-166290
e-mail: msarbu@agora.ro               http://www.agora.ro
-------------------------------------------------------------
```

2. D-na Rinaldi are câteva pagini Web foarte interesante despre netichetă. Adresa o aflați chiar din semnătură.

```
*********************************
Arlene H. Rinaldi - Academic/Institutional Support Services
Florida Atlantic University, Boca Raton, FL - RINALDI@ACC.FAU.EDU
CWIS Coordinator & End User Services (Education/Training)
http://www.fau.edu/rinaldi/arlene.html
```

3. O semnătură care se distinge printr-un desen ingenios realizat cu caractere ASCII (datele sunt fictive).

```
                    )))))
                    (o o)
=====================o00=(_)=00o=================================
*  Bogdan Marinescu                 Fictiv Representative Office   *
*  Marketing & Communications                in Romania            *
*       Tel: (+40 1) 320.23.53.          Bloc High, 12th floor,   *
*       Fax: (+40 1) 320.14.53.          Bd. N.Balcescu 24-26,    *
*    E-mail: fmarines@fictiv.ro          Bucharest 1, Romania     *
=================================================================
```

4. O semnătură ceva mai austeră, care atrage în schimb atenția printr-un citat din Lao Zi și prin faptul că își oferă, la cerere, cheia PGP publică celor care utilizează acest mod de cifrare.

```
+-----------------------------------------------------------------+
|   Dan Popa                  "Who stands up on his tips is not stable|
| telecomms engineer            on his feet" (Lao Zi, "Dao De Jing") |
|email: dpopa@hotmail.com      [PGP public key available on request] |
+-----------------------------------------------------------------+
```

Liste de discuții

Posibilitatea firească a poștei electronice de a trimite simultan copii ale unui mesaj la mai multe adrese a condus la ideea așa-numitelor „liste de discuții" (*mailing lists*), orientate pe domenii, care grupează în jurul lor utilizatorii interesați de tematica pusă în discuție. Unele liste au o tematică foarte bine definită (de pildă utilizarea unui anumit produs, cercetările dintr-un anumit domeniu științific etc.), altele grupează doar persoane care au anumite interese sau doar pasiuni comune (de pildă posesorii de câini dintr-o anumită rasă, vorbitorii unei anumite limbi, membrii unei comunități, veteranii unui anume război etc.).

Principiul pe care se bazează listele de discuții este simplu: un mesaj adresat listei (termenul consacrat pentru această acțiune este „a posta", echivalent oarecum cu „a publica", „a face public") este distribuit în mod automat tuturor membrilor listei. Pentru a deveni membru al unei liste trebuie să vă abonați (*to subscribe*) la respectiva listă. Puteți să abandonați o listă prin operațiunea inversă (*to unsubscribe*). Fiecare listă are o descriere a tematicii abordate și regulile ei de conduită. Listele pot să fie controlate de un moderator sau pot să fie libere.

Administrarea unei liste este realizată de programe specializate, dintre care cel mai răspândit este *Majordomo*. În afară de faptul că distribuie mesajele postate unei liste către abonați, aceste programe răspund la mesajele care le sunt adresate, procesând comenzile pe care le identifică (*subscribe* și *unsubscrie* sunt doar două dintre ele).

Netichetă

Cuvântul „netichetă" (în engleză *netiquette*) a fost compus din *Net* (Rețea) și „etichetă" (reguli de comportare civilizată). Semnificația sa este deja clară: desemnează un set de convenții pe care e bine să le respectăm în utilizarea Internetului în general și a poștei electronice în special.

În primul rând trebuie spus că toate normele unei corespondențe civilizate sunt implicit valabile și în

Clienți de e-mail

Poșta electronică a pătruns, insidios, în viața noastră de zi cu zi. Am devenit dependenți de ea într-un mod mai mare decât ar părea la prima vedere. Cui nu i s-a întâmplat să fie silit să petreacă o perioadă (ore sau chiar zile pentru cei nenorocoși) fără posibilitatea de a trimite sau primi răvașe pe această cale? Abia în acele clipe ne dăm seama cît de mult depindem de fluxul informațional care ajunge la noi prin e-mail; spre deosebire de alte canale prin care suntem bombardați cu informații, acesta are avantajul de a fi biunivoc, deci mai complex. De activitatea acestui canal informațional trebuie, prin urmare, să se ocupe o aplicație complexă și ea - mult mai complexă decât ar părea la prima vedere. Evident, este vorba de clientul de e-mail.

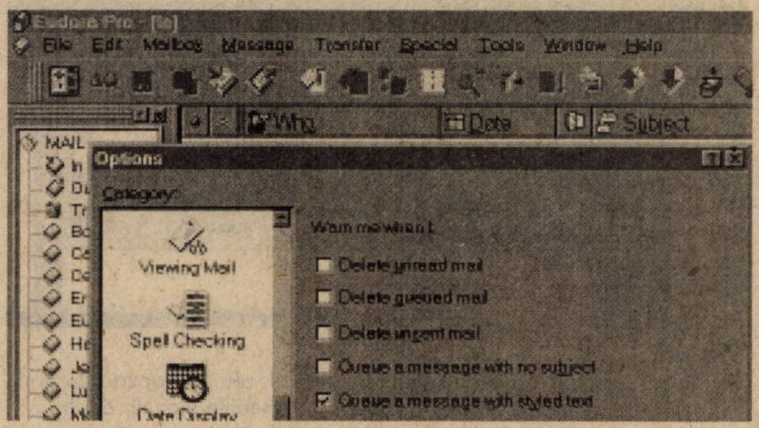
La testare au participat patru programe: două celebre (Eudora Pro și Pegasus) și două mai puțin celebre, însă în plină afirmare (Becky și Calypso).

În competiția clienților de e-mail, este greu de dat un câștigător. Alegerea rămîne personală, depinzând atât de necesitățile fiecăruia, cât și de mașina pe care rulează aplicația.

Gabriel Proșcanu

Eudora Pro 4

Eudora 4 este un program complex și arătos. Interfața clasică a versiunii precedente (Eudora Pro 3.0.5) a fost înlocuită de o interfață care se conformează cerințelor vremii - butoane senzitive, posibilitatea de a o customiza etc). Unii pot spune că Eudora a renunțat la personalitatea proprie, în favoarea uneia care seamănă mult cu Netscape Communicator; per total, însă, impresia vizuală este favorabilă.



Plusuri:

- poate citi și crea mesaje HTML; dacă IE 4.x este instalat, profită de viewer-ul acestuia pentru a vizualiza mesaje foarte complexe (applet-uri, tabele etc.);
- are posibilitatea de a edita mesajele, pentru a economisi spațiul ocupat pe disc;
- filtre excelente; se pot crea condiții de tranzacționare a mesajelor direct pe server, fără să mai fie nevoie descărcarea acestora;
- implementează MAPI, având server propriu;
- beneficiază de PGP, prin intermediul unui plug-in.

Minusuri:

- încă nu are multithreading! în timpul în care trimiteți/primiți mesaje sau le filtrați, nu aveți altceva de făcut decât să vă uitați cum lucrează...
- este mai lentă decât versiunile anterioare (mărimea executabilului s-a dublat), iar stabilitatea sa încă nu este atât de mare ca la acestea;
- un fișier de help destul de slab, mai ales în ceea ce privește editarea fișierului Eudora.ini, cu toate posibilitățile oferite.

rețea. Neticheta tratează doar aspectele specifice corespondenței în Internet. Voi începe cu câteva indicații privind scrierea unui mesaj:

• Aliniați întotdeauna rândurile la marginea din stânga. Nu încercați să centrați titlul sau formula de adresare, deoarece nu puteți prevedea ce efect va avea aceasta pe ecranul destinatarului.

• Lăsați câte un rând gol între paragrafe. Nu folosiți *tab*-uri și nu indentați primul rând al paragrafului.

• Nu scrieți mai mult de 60-70 de caractere pe un rând și nu introduceți caractere de control.

• Chiar dacă editorul vă permite, nu formatați textul. Este posibil ca destinatarul să folosească un terminal alfanumeric sau un mailer care nu recunoaște formatarea, este posibil ca fonturile pe care le folosiți să nu fie disponibile pe calculatorul acestuia sau culoarea textului să fie tocmai cea aleasă pentru fundal...

• Semnați întotdeauna mesajul, mai ales când îl postați într-o listă. E posibil ca nu toți cei ce primesc mesajul să vă cunoască.

• Semnătura nu trebuie să fie prea lungă (nu mai mult de 5 - 6 linii) și trebuie să includă numele complet. Mai pot fi incluse în semnătură numele organizației prin care beneficiați de acces Internet, adresa paginii Web (personale sau a organizației), adresa poștală, număr de telefon și fax (vezi caseta „Semnături").

• Limitați-vă la tratarea unui singur subiect într-un mesaj și completați în mod pertinent câmpul *Subject*, astfel încât destinatarul să poată să identifice cu ușurință mesajele. Dacă aveți mai multe subiecte, scrieți câte un mesaj pentru fiecare.

• Nu scrieți cuvintele cu majuscule decât, eventual, pentru a sublinia un aspect foarte important sau pentru titluri. Scrierea cu majuscule este interpretată ca RĂCNET!

• Pentru a sublinia sau a accentua un anumit *cuvânt*, încadrați-l între asteriscuri. Citarea _titlului unei cărți_ sau a unei lucrări se face între liniuțe de subliniere (*underscore*).

• Scrieți explicit datele calendaristice, deoarece pe glob există diferite convenții și se poate ajunge ușor la confuzii. Deci *11 Mar 1998* și nu *11.03.98* (americanii vor înțelege 3 noiembrie).

• Evitați exprimările dure sau vulgare. Mesajele sunt ușor de trimis mai departe...

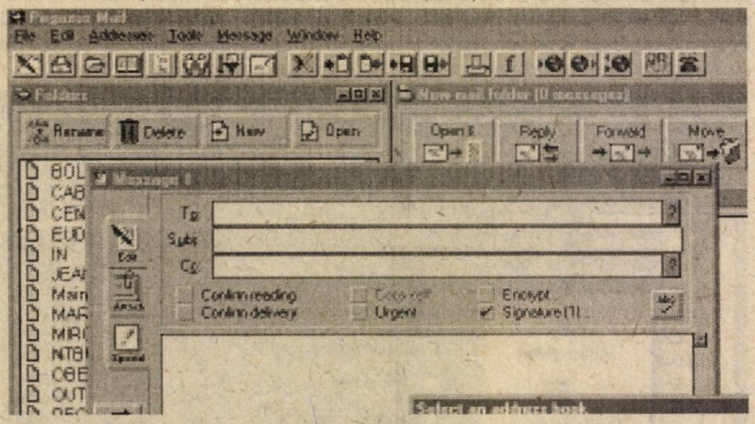
• Nu postați niciodată un mesaj privat într-o listă de discuții, decât cu aprobarea autorului.

• Nu cereți „confirmare de citire" (atenție, e altceva decât confirmare de livrare), chiar dacă mailer-ul vă oferă această posibilitate. Este considerată o intruziune în intimitatea corespondentului. Este mai politicos să cereți politicos destinatarului să confirme primirea mesajului.

• Atenție la glume și sarcasm. Este posibil ca în absența mimicii, intonației și a ceea ce se cheamă „limbajul trupului" (*body language*) destinatarul să nu înțeleagă corect. De asemenea, e posibil ca fondul cultural sau religios diferit al destinatarului să-l conducă la o interpretare pe care nu ați vizat-o. Pentru siguranță, folosiți semnele

Pegasus 2.5.5

Acest program, o adevărată bijuterie de programare (scris de un singur om!), este gratuit! Fișierul său de help este foarte complex și oferă răspunsuri la orice întrebare. Din păcate, interfața sa nu este conformă standardelor impuse de Windows 95, amintind mai mult de Windows 3.1 Comunitatea utilizatorilor Pegasus așteaptă, cu nerăbdare apariția versiunii 3.0 (promisă pentru sfârșitul anului trecut...) care să aducă această compliantă.



Plusuri:

- viteza! este cel mai rapid client de e-mail, atât la tranzacțiile prin rețea (beneficiază de optimizări pentru mediile Novell, dacă acestea sunt detectate), cât și la operațiile interne;
- filtrele extrem de complexe; nu numai că se pot crea structuri beneficiind de operatori logici (AND,OR,XOR,NOT), dar se pot rula și aplicații exterioare, ca urmarea a condițiilor de filtrare; în plus, se pot crea anumite reguli pentru un folder dacă acesta este deschis și alte reguli dacă acesta este închis, în momentul aplicării acestora;
- are un cititor de digest-uri integrat (singurul program, din cele testate, care oferă această opțiune importantă), care tratează digest-urile ca pe o simplă colecție de mesaje, în interiorul căreia se pot face operațiuni de sortare, ștergere etc. În plus, utilizatorul are și posibilitatea de a crea propriile digest-uri din mesajele individuale alese de el;
- suportă PGP (prin add-ons).

Minusuri:

- o interfață „dinozaurică", în care clicul-dreapta este absent;
- dificultăți la selectarea și copierea cu Copy/Paste; mesajele primite nu pot fi editate;
- nu suportă mesajele HTML, pe care le vede în browser sau ca simplu text, pierzând formaterea.

cunoscute sub numele de *smileys* (în traducere liberă: „zâmbilici") sau *emoticons*. Iată în continuare câteva dintre cele mai des utilizate:

:-) – *ha, ha... e o glumă, n-o lua prea în serios*. Se folosește pentru a marca o remarcă glumeață sau (ușor) sarcastică.

:-(– *din nefericire...* Exprimă supărare sau dezamăgire,

;-) – *te-ai prins, nu?* Asemănator primului, dar implică o doză de complicitate (ca și cum ai trage cu ochiul).

:-> – *I-am zis-o!* Marchează o remarcă răutăcioasă, sarcastică.

:-I – *Mda...* Exprimă indiferență, plictiseală.

Câteva sfaturi utile pentru prima participare într-o listă de discuții:

• Citiți descrierea listei și a regulilor de conduită stabilite de moderatorul listei sau de comunitatea participanților. Conformați-vă.

• Familiarizați-vă cu participanții, tematica și „tonul" general citind arhivele ultimelor săptămâni, eventual ca *digest* (formă prescurtată a arhivei, conținând doar intervențiile considerate importante sau semnificative).

• Feriți-vă de un limbaj dur și de atacuri personale. Cu siguranță vi se va răspunde cu aceeași monedă.

• Scrieți îngrijit și respectați neticheta. Atenție: dacă ați gafat și v-ați dovedit un *newbie*, comunitatea virtuală a listei nu va rata ocazia de a vă aplica o „corecție". E aproape un ritual... pe care-i mai bine să-l evitați.

Am lăsat pentru final cele mai importante trei indicații:

• Concepeți mesaje scurte și la obiect. Timpul costă bani, iar dacă cel care primește mesajul se plictisește, printr-un simplu clic îl poate șterge.

• Verificați de trei ori textul înainte de a trimite mesajul. Greșelile de ortografie vă pot păta iremediabil cartea de vizită.

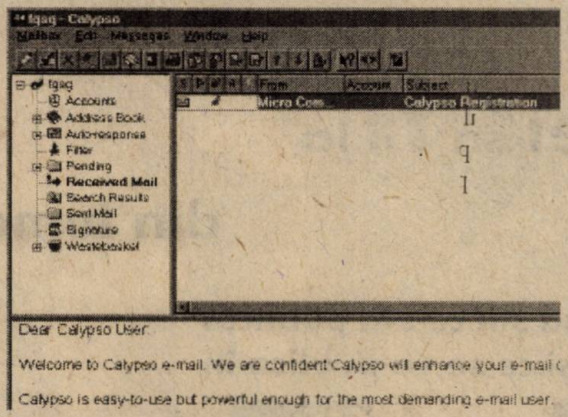
• Nu trimiteți niciodată un mesaj a cărui publicare v-ar putea pune în situații jenante.

❑

Mircea Sârbu este directorul editorial al editurii Computer Press Agora. Poate fi contactat prin e-mail la adresa msarbu@agora.ro.

Calypso 2.4

Deși foarte lăudat, acest mailer este foarte primitiv. Deficiențe serioase îl îndepărtează de utilizatorul obișnuit, care nu beneficiază de singurul avantaj pe care îl oferă.



Plus:

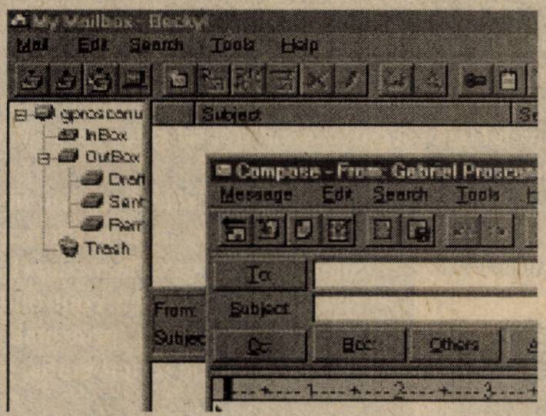
- foarte potrivit pentru *bulk-mailing* (aka SPAM!); își creează o listă cu adrese și poate trimite mii de mesaje foarte ușor.

Minusuri:

- filtre slabe;
- editor de mesaje foarte primitiv (nu acceptă *word-wrap*!)
- căsuțele poștale sunt păstrate într-un format proprietar, cifrat, care face imposibilă recuperarea lor în caz de accident al discului;
- nu citește mesaje HTML;
- nu oferă nici o funcție complexă;
- nu oferă MAPI.

Becky! 1.23

Un program foarte interesant, scris în Japonia (...), care oferă aproape toate facilitățile cerute unui client de e-mail. La fel ca și Pegasus, este scris de un singur om; nu este, însă, gratuit, fiind proprietatea unei corporații.



Plusuri:

- viteza de reacție crescută, atât la tranzacțiile în rețea, cât și în operațiile interne;
- interfață foarte intuitivă;
- poate citi mesaje HTML, folosind ActiveX , fără a lansa browser-ul;
- editor de mesaje foarte sofisticat (un adevărat editor de texte);
- interfață proprie pentru PGP;
- utilitar propriu pentru importarea mesajelor din Eudora, Netscape etc.

Minusuri:

- nu implementează MAPI;
- nu poate crea mesaje HTML;
- interfața proprie nu lucrează decât cu PGP 2.6.3;
- filtrele, destul de complexe, nu sunt încă la nivelul celor din Pegasus sau Eudora.

ESTI PREGĂTIT PENTRU MÂINE ?
ARE YOU READY FOR TOMORROW ?

COMPUTER PRESS AGORA

Vrei să fii la curent cu ultimele știri din lumea IT?

Vrei să știi ce s-a publicat și ce se va publica în

BYTE ROMÂNIA · CAD report · PC REPORT CALCULATOARE PERSONALE · info gazeta info

Conectează-te la www.agora.ro

Prima carte românească de referință în domeniul rețelelor de calculatoare

Autor, profesor și cercetător, laureat al premiului ACM Karl V. Karlstrom Outstanding Educator Award, Andrew S. Tanenbaum, explică cu grijă felul în care rețelele funcționează în interior, pornind de la tehnologia hardware până la cele mai utilizate aplicații de rețea. Cartea urmează o abordare structurată a rețelelor de calculatoare, începând de jos (de la nivelul fizic) și parcurgând treptat calea spre vârful acestora (nivelul aplicațiilor).

Subiectele abordate includ:

- Nivelul fizic (firele de cupru, fibrele optice, comunicațiile radio și cele prin satelit)
- Nivelul legătură de date (principiile protocoalelor, HDLC, SLIP și PPP)
- Subnivelul de acces la mediu (LAN-uri IEEE 802, punți, noile LAN-uri de mare viteză)
- Nivelul rețea (rutarea, controlul congestiei, interconectarea rețelelor, IPv6)
- Nivelul de transport (principiile protocoalelor de transport, TCP, performanța rețelei)
- Nivelul aplicație (criptografia, poștă electronică, știri, Web-ul, Java, multimedia)

În fiecare capitol sunt descrise în detaliu principiile de bază, urmate apoi de exemple ample luate din Internet, rețele ATM și rețele de comunicații fără fir.

Doresc să cumpăr cartea Rețele de calculatoare

V-am expediat în contul C.P. Agora nr.: 2021400027100178 deschis la Banca Transilvania, filiala Mureș, 70.000 lei (în care nu sunt incluse taxele poștale), cu mandatul poștal/ordinul de plată nr.: ..... din data de ..... ☐ Doresc să-mi trimiteți și factură.

Pentru elevi și studenți cartea costă 50.000 lei. Ei vor trebui să atașeze comenzii o copie după carnetul de elev sau student.

Numele și prenumele: ..... Funcția: .....

Firma: ..... Adresa: .....

Cod poștal/Localitate: ..... Telefon/Fax: .....

Data ..... Semnătura .....

Vă place GInfo ?

Nume: Prenume: Vârsta:

Adresa:

Localitate: Cod poștal: Județ: Telefon:

Ocupația: elev informatică ☐ elev ☐ student ☐ cadru didactic ☐ altă ocupație ☐ (precizați:)

Școala (firma, organizația etc.):

Dotări

Folosiți un calculator acasă? Da ☐ Nu ☐

Câte calculatoare sunt instalate în școală: (aprox.)

Aveți acces la e-mail: Da ☐ Nu ☐

Aveți acces la WWW: Da ☐ Nu ☐

Cum apreciați dotarea cu tehnică de calcul a școlii:

Foarte bună ☐ Bună ☐ Slabă ☐ Foarte slabă ☐

Aprecieri generale

Acordați note de la 1 la 10 pe următoarele criterii:

Structura tematică:

Calitatea articolelor:

Prezentare grafică:

Calitate tipografică:

Considerați că prețul este:

prea mare ☐ accesibil ☐ prea mic ☐

De cât timp cunoașteți Gazeta de Informatică (ani)?

Cum ați procurat acest număr:

chioșc ☐ abonament ☐ cadou ☐ altfel ☐

În acest număr

Apreciați cu note de la 1 la 10 articolele acestui număr:

Structuri înlănțuite în Java

Tipuri de date

Kasparov vs. Deep Blue

LOGO

The Never Ending Story

E-mail

Articolul care v-a plăcut cel mai mult:

De ce?

Articolul care v-a plăcut cel mai puțin:

De ce?

Considerați că ponderea problemelor rezolvate este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea problemelor propuse este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea subiectelor teoretice este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că nivelul de accesibilitate al articolelor este

prea cobotât ☐ potrivit ☐ prea ridicat ☐

În viitor

În următoarele numere așteptați:

mai multe articole de programare? Da ☐ Nu ☐

mai multe articole teoretice? Da ☐ Nu ☐

mai multe articole despre Internet? Da ☐ Nu ☐

mai multe prezentări de programe? Da ☐ Nu ☐

prezentări de jocuri? Da ☐ Nu ☐

articole despre hardware? Da ☐ Nu ☐

V-ar plăcea să citiți:

pagini redactate de elevii unor licee? Da ☐ Nu ☐

lucrări premiate, realizate de elevi? Da ☐ Nu ☐

actualități școlare? Da ☐ Nu ☐

prezentarea unor situri Web? Da ☐ Nu ☐

tips & tricks? Da ☐ Nu ☐

Comentarii:

Dacă nu vă este suficient spațiul, folosiți coli albe suplimentare.

Așteptăm scrisorile voastre!

Poșta redacției

Dialogul cu cititorii va fi una dintre preocupările constante ale redacției. Pentru ca acest dialog să se desfășoare în cele mai bune condiții, ne-am propus ca în acest număr să stabilim cadrul general.

Înainte de toate țin să le mulțumesc tuturor celor care mi-au comunicat opiniile și sugestiile lor în privința noii formule de apariție a Gazetei de Informatică (pe scurt, *GInfo*). Vă aștept în continuare mesajele.

Votul tău contează

Pentru ca revista să răspundă cât mai bine așteptărilor voastre, redactorii ei trebuie să vă afle părerile. Forma, structura, tematica, ponderea diverselor rubrici, tot ce ține de *GInfo* reprezintă subiecte deschise și se pot modifica dinamic, în funcție de părerile și sugestiile voastre. Și pentru că, desigur, vom încerca să satisfacem cât mai multe gusturi, va trebui să ne bazăm și pe niște statistici. În acest sens, vă rugăm ca fiecare corespondență către *GInfo* să fie însoțită de chestionarul din pagina precedentă. Completat, desigur.

Adresa redacției este: *GInfo, C.P. 172-1, 3400 Tg-Mureș*. Dacă ne scrieți prin e-mail, solicitați forma electronică a chestionarului trimițând pe adresa:

```
office@ginfo.ro
```

un mesaj având ca *subject* cuvântul „chestionar".

Cele mai bune idei

Evident, majoritatea decide în cele din urmă. Dar asupra ce anume decide? Ideile bune, ideile grozave (*cool* - dacă vă place mai mult) sunt apanajul exclusiv al individului. Dacă vă vine vreuna cu privire la *GInfo*, nu stați pe gânduri: pac, la poșta redacției. Dacă e interesantă, o supunem discuției. Dacă e irezistibilă, o aplicăm imediat. În plus, mai acordăm la fiecare număr și un mic premiu (o carte, un CD, un tricou..) cititorului care ne-a sugerat cea mai originală, cea mai trăznită idee (sau, dimpotrivă, cea mai pragmatică, cea mai serioasă...). Adresele sunt aceleași (terestru și e-mail).

Să discutăm

Cu siguranță, „poșta redacției" nu va putea să găzduiască toate opiniile, toate ideile, toate criticile și toate răspunsurile. Pentru cei care au acces la e-mail (sper că foarte curând, toți) am conceput o formulă mult mai dinamică de dialog: o listă de discuții (*mailing list*) care se va chema chiar „dialog". Pentru a vă abona, trimiteți la adresa:

```
majordomo@ginfo.ro
```

un mesaj care să conțină lina:

```
subscribe dialog
```

Alături de confirmare, veți primi o descriere a listei și a regulilor, după care veți putea participa la discuțiile despre *GInfo*. Mesajele pentru listă se trimit pe adresa `dialog@ginfo.ro`.

Probleme, probleme

Un alt subiect permanent în *GInfo* îl vor constitui problemele propuse spre rezolvare. Cu siguranță, în viitorul an școlar, concursul rezolvitorilor va avea un regulament clar, se va desfășura de-a lungul întregului an școlar (adică 8 numere *GInfo*) și se va „solda" cu premii anunțate din timp. Anul acesta nu putem să anunțăm dinainte anumite premii, dar vom lansa totuși, în mod experimental, concursul. Deși va dura doar trei luni, speranța noastră este că în acest timp vom putea să identificăm și să rezolvăm disfuncționalitățile care ar putea să apară, astfel încât, din toamnă, totul să fie OK.

Așadar, așteptăm rezolvări la problemele propuse. Problemele vor fi identificate printr-un cod (în acest număr: de la P019801 până la P019813). Rezolvările trebuie să fie clare, iar programele să funcționeze. Fiecare rezolvare perfectă valorează 10 puncte, iar punctele se cumulează pentru problemele propuse până în numărul din aprilie (inclusiv). Câștigătorii vor fi anunțați în numărul din iunie.

Soluțiile problemelor din acest număr trebuie să fie expediate până în data de 1 martie (inclusiv) pe adresa redacției, de preferat prin e-mail la adresa `office@ginfo.ro`. Mesajele trebuie să aibă ca subiect codul problemelor rezolvate. Nu uitați să vă scrieți numele înaintea fiecărei soluții și numele școlii, deoarece concursul este doar pentru elevi.

Să colaborăm

Primim cu plăcere spre publicare materiale trimise de cititori. Articolele trebuie trimise în formă electronică, ca documente RTF (*Rich Text Format*). Pentru a fi publicate, ele trebuie să fie originale, să corespundă calitativ (atât din perspectiva conținutului cât și a exprimării) și să se încadreze în profilul revistei. Este indicat să contactați în prealabil redacția pentru a vă asigura de interesul acesteia pentru tematica pe care o tratați.

În afară de articole propriu-zise, primim cu plăcere și alte materiale: propuneri de probleme originale, soluții interesante, relatări despre evenimente care privesc domeniul învățământului de informatică etc. Nu promitem că le vom publica pe toate, dar cu siguranță ne interesează.

În curând pe Web

În luna februarie va lua naștere și pagina WWW a revistei. O veți găsi la adresa `www.ginfo.ro`. Pe parcursul primei luni pagina va fi mai mult *„under construction"*, dar toate lucrurile trebuie să aibă un început...

❑

(M.S.)

Eşti pregătit pentru mâine?

COMPUTER PRESS AGORA

Abonează-te şi vei fi!

Completaţi talonul de mai jos sau cartea poştală din revistă

Numele (Firma): _____

Prenumele: _____

Adresa: _____

Cod poştal/Localitate: _____

Tel./Fax: _____

Am avut abonament în 1997 Da ❑ Nu ❑

V-am expediat _____ lei, reprezentând contravaloarea abonamentului la revistele _____

_____ pe o perioadă de _____ luni, cu mandatul poştal/ordinul de plată nr. _____

din data _____, pentru C.P. Agora, în contul: 2021400027100178, Banca Transilvania, filiala Mureş.

❑ Doresc să-mi trimiteţi factură

Data _____ Semnătura _____

		Preţ vânzare	Preţ abonament	Abonament elevi/studenţi*	Luna din care doriţi să înceapă abonamentul
BYTE România	12 luni	120.000 lei	96.000 lei ❑	72.000 lei ❑	
	6 luni	60.000 lei	48.000 lei ❑	36.000 lei ❑	
PC Report	12 luni	96.000 lei	76.800 lei ❑	57.600 lei ❑	
	6 luni	48.000 lei	38.400 lei ❑	28.800 lei ❑	
CAD Report	6 apariţii	48.000 lei	38.400 lei ❑	28.800 lei ❑	
	3 apariţii	24.000 lei	19.200 lei ❑	14.400 lei ❑	
Ginfo	5 numere**	25.000 lei	25.000 lei ❑	20.000 lei ❑	

*Elevii şi studenţii vor trebui să ataşeze comenzii o copie după carnetul de elev sau de student

** În anul şcolar 1998

HEWLETT® PACKARD

Seria de imprimante HP DeskJet 690 cu PhotoREt. Imaginea afacerilor rentabile.

Legenda spune că o imprimantă nu poate însufleți imaginile. Datorită unei strălucite combinații de tehnologii avansate, seria de imprimante HP DeskJet 690 spulberă odată pentru totdeauna această prejudecată. Imaginile imprimate au calitate fotografică datorită tehnologiei HP PhotoREt. Sistemul RealLife imaging asigură, cele mai vii culori şi cea mai bună calitate pentru reproducerile alb-negru. Mai mult decât atât! ColorSmart reglează automat nivelurile de culoare. Cu cartuşul Photo Cartrige, se obțin cele mai naturale culori, iar imaginile tale beneficiază de o calitate fotografică excepțională şi de o mai mare rezoluție. Nici n-apucă bine balaurul să pocnească din coadă şi documentele tale sunt deja tipărite, 5 pagini alb/negru sau 2 pagini color pe minut! Imprimanta HP DeskJet 690C reproduce imagini atât de reale de parcă simți arsura flăcărilor scoase de balaur. Aşa că, uită poveştile! Seria DeskJet 690 nu este una oarecare. Este sufletul documentelor tale!

IMPRIMANTELE HP. HÂRTIA ESTE ÎN SLUJBA TA.

DEALERI – Bucureşti: AMERILEX - 223 43 23 • MBL COMPUTERLAND - 336 75 99 • Q'NET - 211 78 55 • S&T ROMANIA - 330 73 20 • SYSTEM PLUS - 210 85 70 THEMA - 231 29 55 • **Bistrița:** LOGITEC COMPUTER - 063/23 27 96 • **Cluj:** SISTEC-CLUJ - 064/19 02 82 • **Timişoara:** BRIDGEMEN - 056/20 13 60
RESELLERI – Bucureşti: ALSYS DATA - 411 27 27 • BLUE RIDGE INTERNATIONAL - 210 68 28 • CIPRO - 220 04 00 • FLAMINGO COMPUTERS COM - 222 49 34 • FOCUS INTERNATIONAL - 322 30 69 NET CONSULTING - 638 33 45 • PRODUCTON - 312 30 78 • RADIX - 410 25 25 • SOFTWARE ITC - 232 44 71 • TRICORP ELECTRONICS - 679 48 79 • TOTAL SOFT - 230 09 76 **Baia Mare:** CUMULUS LOGIC - 062/43 07 35 • **Bacău:** MIDO IMPEX - 034/17 10 80 • **Braşov:** ICCO COMPUTERS - 068/14 43 54 • INTELPROF INTERNATIONAL - 068/41 13 54 SISTEM - 068/31 86 63 • **Cluj Napoca:** AT COMPUTER - 064/41 32 04 • **Craiova:** AS COMPUTERS - 051/16 32 80 • **Constanța:** FORTE DIGISOFT - 041/65 30 36 • GMB COMPUTERS - 041/61 92 22 **Deva:** ASCOPIER - 054/23 36 06 • **Galați:** ROMLOTUS - 036/47 40 00 • SPARK - 036/46 52 70 • **Hunedoara:** MEMORII TECH DATA SERVICE - 054/23 07 03 • **Iaşi:** RADIX - 032/ 21 24 53 • SERCOM - 032/12 25 89 **Piatra Neamț:** SOFT COMPREST - 033/ 22 21 21 • **Ploiești:** PLATIN SISTEMS - 044/19 18 18 • **Reşița:** ELCOM - 055/22 07 11 • **Suceava:** PRONOVA SMART - 030/52 14 85 **Timişoara:** ETA-2U: 056/19 95 08 • NETRING COMPUTER - 056/13 28 94 • **Tg. Mureş:** ELECTRO ORIZONT - 065/16 65 72 • **Tulcea:** MICROLINE - 040/51 63 69
CENTRU DE SERVICE AUTORIZAT: 330 73 00

Martie 1998
Vol. 8/2

Lei 5000

Gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

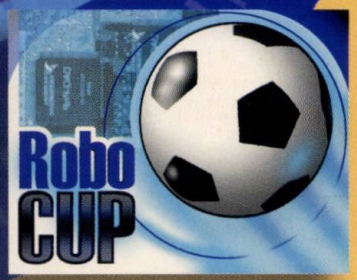
din sumarul numărului:

MAI REPEDE, MEREU MAI REPEDE...

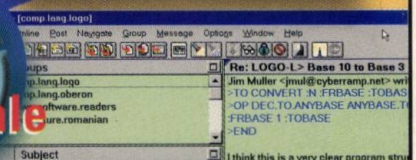
Durata de execuție a algoritmilor

Algoritmul Boyer-Moore

Frontiere:
Fotbal, software și roboți

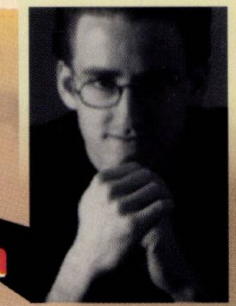


Internet:
Rețeaua și comunitățile virtuale



Serial:
A doua lecție de Java

Portret:
1st Person:
cei care au creeat Java



Plus:
Probleme și rezolvări, interviu, puțină matematică...
Și altele.

T.P.
Aprobat R.A.P.R.
Nr.103/DC/053/1998
Valabil - 1998 -

INFADRES

aveți
• un domeniu de activitate
• mai multe birouri/sedii
• angajați
• multe hârtii și probleme de rezolvat
HARABABURĂ !!!
vă doriți
• echipamente hardware interconectate
• software de interfață & de aplicație
• școlarizarea angajaților
ORDINE !!!
vă oferim
• soluții hardware & software
• sisteme integrate "la cheie"
• asistență tehnică software & service hardware
• consultanță
• training
cartea noastră de vizită
• System Integration & Project Management - proiecte la nivel național
• IBM Business Partner
• Microsoft Authorized Distribution Partner/ DSP for Romania & Microsoft Hot Line
• distribuitor Symantec
• distribuitor de PC-uri FORTE
• 9 sucursale FORTE
Microsoft
WHERE DO YOU WANT TO GO TODAY ?™
Authorized Distribution Partner/ DSP for Romania
FORTE
COMPUTERS
Flexibility.Made in Romania
IBM
Lipscani 102, București
Tel: 3122360/6137282 Fax: 3122630 Microsoft Hot Line: 3120948

Vol. 8, Nr. 2, Martie 1998

gazeta de informatică

Σ Sumar

soluții.. PAG. 5

probleme REZOLVATE
pagini realizate de CLARA IONESCU

Încă patru probleme de la Olimpiada Națională, Timișoara, 1997.

probleme.. PAG. 15

Propuse pentru REZOLVARE
***

Probleme propuse la concursul memorial "Nemes Tihamer" - 1998

interviu.. PAG. 40

MONICA de la MICROSOFT
de MIRCEA SÂRBU

La 25 ani de ani a terminat două facultăți și lucrează la Microsoft.

focus.. PAG. 19

Algoritmul BOYER-MOORE
de COSMIN TRUȚĂ

O metodă de căutare rapidă a unui șir.

PAG. 22

DURATA de execuție a ALGORITMILOR
de RALUCA VERNIC

Sunt oare eficienți algoritmii pe care-i concepem? Putem afla răspunsul.

frontiere.. PAG. 26

Fotbal, SOFTWARE și roboți
DE IONUȚ MUȘLEA

Inteligența artificială își spune cuvântul și în sportul rege.

serial.. PAG. 30

JAVA — a doua LECȚIE
de PROF. DR. HORIA GEORGESCU

Se poate face oare în Java tot ce se poate face în Pascal sau C?

profil.. PAG. 44

CUM s-a născut JAVA
de GABRIEL PROȘCANU

Povestea unui grup de oameni care-au căutat ceva și-au găsit altceva.

internet.. PAG. 51

NEWS: informație și OPINIE
de MIRCEA SÂRBU

În forumurile virtuale din Rețea se publică zilnic o enciclopedie.

mate.. PAG. 48

Problema lui STEINHAUS și Sierpinski
de ILIE VIERU

Redacția:

Director științific:
Prof. dr. *Horia Georgescu* (București)
Redactor șef:
prof. *Clara Ionescu* (Cluj)
Editor executiv:
Mircea Sârbu (Tg-Mureș)

Au mai colaborat:
Ionuț Mușlea (Cluj)
Ilie Vieru (Bacău)
Dragoș Marian (Tg-Mureș)
Gabriel Proșcanu (București)
Raluca Vernic (Constanța)
Cosmin Truță (Cluj-Napoca)

Tehnoredactare:
Călin Borz
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
C.P. 172-1, 4300 Tg-Mureș
Adresa e-mail: office@ginfo.ro
Tel: (065) 166.516
Fax: (065) 166.290
http://www.ginfo.ro

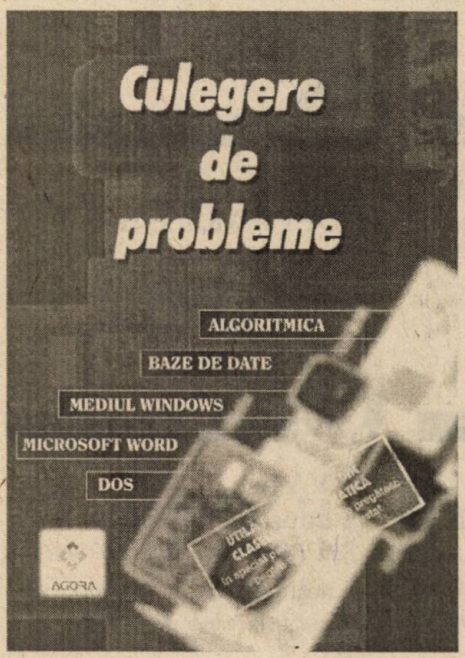
ISSN: 1222-7129
Tiparul: *S.C. Infopress srl*, Odorheiu Secuiesc, tel: 066.216.483

Editor: Computer Press Agora srl
str. T. Vladimirescu nr. 63/1
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director editorial:
Mircea Sârbu
Director tehnic:
Szabó László
Reprezentanța București:
Director: *Dana Pastramă*
Tel: (01) 330.92.81

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Culegerea conține:

- probleme de algoritmică, majoritatea rezolvate în Pascal
- probleme de gestiune a bazelor de date (FOX-PRO)
- probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word
- probleme de vericare a cunoștințelor aferente sistemului de operare DOS

Această culegere constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru atestatul de informatică dar poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, culegerea conține o varietate mare de enunțuri de probleme rezolvate, abordând diferite metode și stiluri de programare.

• • •

Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) mai elegant și cu o eficiență în programare mai ridicată.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422] sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

DE LA vise la CUVINTE

Și uite așa am intrat în rândul... revistelor "adevărate". Mă și sperie această exprimare, deoarece parcă aud (cu urechile sufletului) remarca — pe drept revoltată — a multora dintre cititori: "Cum adică? *Gazeta de informatică* a fost o revistă... *adevărată* și până acum! Și nu una oarecare, ci una extrem de serioasă...". Da, am fost și vom rămâne serioși; în plus am câștigat în profesionalism, plus care va îmbunătăți imaginea revistei, care va adăuga "culori noi" conținutului și care va ajuta revista să ajungă la toți acei cititori pentru care a fost gândită, scrisă și care o așteaptă.

Nu este deci un secret motivul pentru care s-a schimbat editorul. Meritul echipei anterioare este, în primul rând, că nu a lăsat revista "în paragină", an de an a încercat și a reușit să îmbunătățească și să diversifice conținutul, condițiile de tipar, modul de difuzare, pe scurt "semantica" și "sintaxa". Pe de altă parte, gândindu-mă acum (după "schimbarea la față") la primul număr apărut la Editura Petrion în noiembrie 1991 sau la cel de la Editura Libris (aprilie 1992), constat că a trecut un "secol"! Sună retoric, dar cred că îmi dați dreptate că în acești ani informatica s-a modificat esențial și nu se mai poate respecta în sens strict proiectul gândit și realizat (de foarte multe ori cu efort deosebit) de cei care au lansat primul număr al revistei.

În 1991 am lucrat (în școli) pe 286, am programat în Pascal (oare ce versiune?), ne bucuram de facilități oferite de Norton Commander...

Azi? Azi ne plimbăm pe Internet, ne "muncim" cu programarea orientată pe obiecte, cu procesoare de texte, de imagini...

Rezultă, nu-i așa? A trecut un "secol". Și dacă e așa, înseamnă că și revista viitorilor specialiști în tehnologia informației (vedeți — nici nu mai spun *informatică*) trebuie să se alinieze dinamic la ceea ce ne oferă sau mai bine zis, ce ne cere acest prezent.

Colegial

Îmi aduc aminte cu plăcere de colegi — profesori de informatică — dar și de alții care nu erau neapărat de specialitate informatică și care ne-au ajutat pe parcursul anilor în a face cunoscută revista. Faptul că acum ea se difuzează în alte condiții decât până acum nu înseamnă că nu mai avem nevoie de sprijinul lor. Este o revistă realizată pentru elevi, care își petrec o mare parte din timp în școală. Ar fi bine dacă școala ar observa că orice colaborare cu GInfo, chiar și doar în calitate de cititor, aduce beneficii tânărului care vrea să fie pregătit în mileniul III pentru ceea ce va pretinde acesta de la el.

Cu optimism

Revenind la încrederea cu care privesc spre viitor, afirmând că mulți cititori așteaptă următorul număr al GInfo, îmi imaginez că mă întrebați: de unde atâta siguranță pe noi, cei care o "facem"?

De unde știm noi câți o așteaptă? Deocamdată ne bazăm doar pe câteva informații, cum ar fi de exemplu, numărul acelora care l-au contactat prin e-mail pe Mircea Sârbu, directorul editorial, sau numărul acelora care în urma apariției primului număr la *Computer Press Agora* se interesează de modul în care se pot abona sau de modalitățile de colaborare cu GInfo.

Nu vă mirați, în ziua în care scriu aceste rânduri au sosit deja articole și rezolvări ale problemelor propuse.

Cititorii noștri se bucură de posibilitatea de a-și exprima părerea, de ușile deschise ale revistei și ne caută cu încredere. Sperăm să fim pe măsura așteptărilor și să devenim un sprijin din punct de vedere profesional. Nici nu se poate altfel, având în vedere și apelul nostru: "Ești pregătit pentru mâine?"

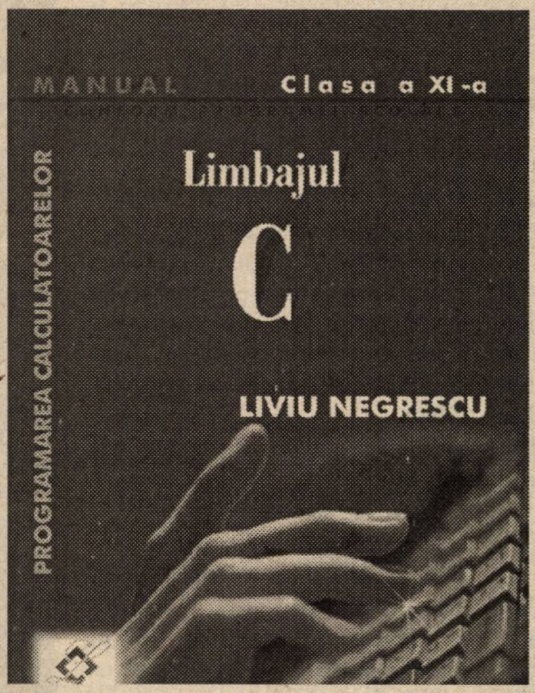
Dar gusturile diferă. De asemenea așteptările cititorilor. Un lucru este sigur: nu vom putea acoperi aria de interes extrem de diversificată a preocupărilor și dorințelor tuturor cititorilor (în primul rând din motive de spațiu!). Și totuși, în ideea de a "configura" profilul revistei, sunteți invitați să contribuiți la formarea lui cu părticica visată. Atenție însă: visele nu ajung, e nevoie de *„hârtie și cerneală"*, sau de o tastatură și e nevoie de cuvinte (care trebuie scrise și rescrise și... iar rescrise) pentru a exprima idei valoroase.

Clara Ionescu

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Această carte este concepută în conformitate cu Programa analitică a clasei a XI-a, profilul Informatică. Ea poate fi considerată o introducere elementară în limbajul C. Fiecare noțiune este explicată în mod clar și utilizată în programe simple. De asemenea, capitolele sunt însoțite de exerciții propuse care contribuie la fixarea și înțelegerea noțiunilor întâlnite în capitolele respective.

Cartea se adresează atât elevilor cât și celor interesați să se inițieze în limbajul C care în prezent este unul dintre cele mai utilizate limbaje de programare.

Liviu Negrescu este cercetător principal gr. I la societatea SOFTWARE ITC S.A., filiala Cluj, unde se ocupă cu dezvoltări de programe de sistem și aplicative.
În același timp, desfășoară o activitate didactică prestigioasă de peste 15 ani, la Facultatea de Calculatoare de la Universitatea Tehnică din Cluj.
În decursul anilor a fost titular a mai multe cursuri, ca de exemplu:

- Limbaje de programare și compilatoare;
- Proiectarea translatoarelor etc.

Liviu Negrescu are o serie de lucrări științifice publicate în diferite reviste de specialitate. În colaborare cu alți cercetători de la SOFTWARE ITC S.A. a implementat sisteme cu aplicații în diferite domenii:

- Construcția de compilatoare;
- Interfețe utilizator;
- Prelucrări de date din domenii economice și medicale.

Liviu Negrescu este autorul mai multor cărți apărute în decursul anilor, cu privire la diferite limbaje de programare (CPL1-subset al limbajului PL/1, CDL-limbaj de descriere al compilatoarelor precum și referitoare la limbajele C și C++. Lucrarea beneficiază de experiența vastă a autorului câștigată atât în activitatea de cercetare cât și în cea didactică.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

SOLUȚIILE problemelor propuse

Numărul 1/1998 al Gazetei de Informatică v-a întâmpinat cu rezolvări ale unor probleme de la Olimpiada Națională de Informatică (1997). În acest număr vă propunem să studiați și să analizați câteva soluții date de autorii problemelor.

Clasa a IX-a: Spărgătorul de ... bănci!

Problemă propusă de prof. Doru Anastasiu, Liceul "Radu Greceanu", Slatina

Enunț:

Clădirea unei bănci cu n niveluri ($1 \le n \le 1000$) are scări de acces pe două părți; acestea sunt notate cu s (stânga) și d (dreapta). Pe scări, la anumite niveluri se află posturi de pază. La fiecare nivel sunt plasate seifuri cu saci de bani (cel mult 100 de saci pe un nivel).

Un hoț pătrunde în clădire prin una din cele două uși de la parter și intenționează să ajungă pe acoperiș, unde este așteptat de un complice cu un elicopter. Evitând cu dibăcie posturile de control, hoțul traversează unele niveluri (într-un singur sens), adunând sacii cu bani din seifurile pe care le întâlnește în cale. Să se stabilească un traseu (dacă există), pe care îl va parcurge hoțul, astfel încât să ajungă la elicopter cu cât mai mulți saci de bani.

Date de intrare:

Datele de intrare se vor citi din fișierul INPUT.TXT, având structura:

n – *numărul de niveluri;*

$x_1\ x_2 \ldots x_p$ – *nivelurile la care se află posturi de control pe partea stângă* ($1 \le p \le n$);

$y_1\ y_2 \ldots y_q$ – *nivelurile la care se află posturi de control pe partea dreaptă* ($1 \le q \le n$);

$s_1\ s_2 \ldots s_n$ – s_i *este numărul de saci din seiful de pe nivelul* i;

Observație: Dacă nu există nici un post de control pe una sau ambele părți, liniile corespunzătoare din fișier vor conține un 0.

Datele de intrare sunt corecte.

Date de ieșire:

Rezultatele vor fi scrise în fișierul OUTPUT.TXT, având structura:

m – *numărul de saci colecționați;*

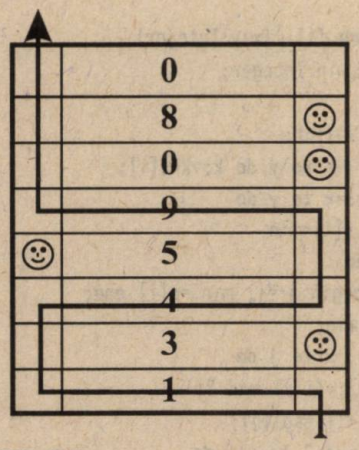
$u_1u_2 \ldots u_nu_{n+1}$ – *partea pe care este atins fiecare nivel* ('s' sau 'd') de la nivelul 1 la nivelul n (u_{n+1} corespunde părții pe care este atins acoperișul).

În cazul în care hoțul nu poate ajunge la elicopter, se va scrie în fișier:

```
0
#
```

Exemplu:

```
INPUT.TXT:
8
4
2 6 7
1 3 4 5 9 0 8 0
```




```
OUTPUT.TXT:
14
dsddssss
```

Soluție:

Datele de intrare se citesc din fișierul INPUT.TXT folosind șirurile st și dr pentru reținerea pozițiilor posturilor de pază, cu componentele 1 sau 0 (1 pentru post de pază și 0 altfel). Numărul de saci cu bani pentru fiecare etaj este reținut în șirul s. Soluția va fi construită în șirul sol cu semnificația sol[i]:=0, i=1..n+1, pentru scara din stânga și 1, pentru cea din dreapta.

În prima parte a algoritmului se verifică dacă problema are soluție, adică dacă există posturi de pază pe același nivel sau pe niveluri consecutive, pe ambele scări. Aflăm primul nivel la care se află un post de pază și se verifică paritatea acestuia pentru a putea alege scara (stânga sau dreapta) cu care se începe traseul (traversând toate nivelurile). Se caută următorul nivel cu post de pază și se continuă traseul; în funcție de paritatea diferenței dintre nivelurile cu posturi de pază se folosește sau nu nivelul care conține cei mai puțin saci ș.a.m.d.

Prezentăm rezolvarea studentului *Vlad Marius*, folosită în programul de verificare la evaluarea programelor realizate de elevi la ONI 1997.

Listing BANCI.PAS

```
Program Spargatorul_de_banci;
var i,li,j,jj,n,m,p,q:Integer;
    k:Longint;
    s,st,dr,sol:array[1..1001] of Integer;
    fi,fo:Text;
procedure fil(x,y:Integer);
var i:Integer;
begin
  for i:=x to y+1 do
  begin
    sol[i]:=(i-x) mod 2;
    if i<=y then k:=k+s[i]
  end
end;

procedure fil_1(x,y:Integer);
var i,j,min:Integer;
begin
  min:=MaxInt;
  for i:=x to y do k:=k+s[i];
  for i:=x to y do
    if s[i]<min
    then
      begin j:=i; min:=s[i] end;
  k:=k-min;
  for i:=x to j do
  sol[i]:=(i-x) mod 2;
  sol[j+1]:=sol[j];
  for i:=j+2 to y+1 do
  sol[i]:=(i-x+1) mod 2
end;
```

```
procedure intrare(x,y,dir:Integer);
var i,j:Integer;
begin
  if sol[x]<>dir
  then
    for i:=x to y do
      if sol[i]=1 then sol[i]:=0
                  else sol[i]:=1
end;

Begin
  Assign(fi,'INPUT.TXT'); Reset(fi);
  Assign(fo,'OUTPUT.TXT'); Rewrite(fo);
  Readln(fi,n);
  for i:=1 to n do st[i]:=0;
  for i:=1 to n do dr[i]:=0;
  while not Seekeoln(fi) do
    begin Read(fi,j); st[j]:=1 end;
  Readln(fi);
  while not Seekeoln(fi) do
    begin Read(fi,j); dr[j]:=1 end;
  Readln(fi);
  for i:=1 to n do Read(fi,s[i]);
  k:=0;
  for i:=1 to n do
    if (st[i]+dr[i]=2) then k:=1;
  for i:=2 to n do
    if (st[i]+dr[i-1]=2) or (st[i-1]+dr[i]=2)
    then k:=1;
  if k=0
  then
    begin
      k:=0; i:=1;
      while (i<=n) and (st[i]+dr[i]=0) do i:=i+1;
      fil(1,i-1);
      if st[i]=1 then jj:=0
                 else jj:=1;
      if ((i mod 2=1) and (jj=0)) or
         ((i mod 2=0) and (jj=1))
      then intrare(1,i,1);
      sol[i+1]:=sol[i];
      li:=i+1;
      while li<=n do
      begin
        i:=li;
        while (i<=n) and (st[i]+dr[i]=0) do
          i:=i+1;
        if i>n
        then
          begin j:=sol[li]; fil(li,n);
            intrare(li,n+1,j)
          end
        else
          begin
            if st[i]=1 then jj:=1
                       else jj:=0;
```

```
            j:=sol[li];
            if (((i-li) mod 2=0) and (j=jj)) or
               (((i-li) mod 2=1) and (j<>jj))
            then
              begin
                fil(li,i-1);
                intrare(li,i,j);
                sol[i+1]:=sol[i]
              end
            else
              begin
                fil_1(li,i-1);
                intrare(li,i,j);
                sol[i+1]:=sol[i]
              end
          end;
        li:=i+1
      end;
    Writeln(fo,k);
    for i:=1 to n+1 do
      if sol[i]=0 then Write(fo,'s')
                  else Write(fo,'d');
    Writeln(fo)
    end
  else
    begin
      Writeln(fo,0); Writeln(fo,'#')
    end;
  Close(fi);
  Close(fo)
End.
```

Clasa a X-a: Avenul SPEOTIM-97

Problemă propusă de prof. Rodica Pintea, Liceul "Grigore Moisil", București

Enunț:

Institutul de cercetări speologice din România a anunțat descoperirea recentă în zona Munților Bihorului a unui sistem subteran de coridoare (culoare orizontale) situate pe diferite niveluri. Coridoarele comunică între ele prin puțuri (verticale). Fiecare culoar vertical se termină într-un coridor (culoar orizontal) care are două capete (diferite de punctul unde ajunge puțul). La capătul unui coridor poate fi un alt puț sau o fundătură. Între cele două capete ale unui coridor nu există alte ramificații.

Tinerii de la *SpeoTim* au pornit într-o expediție de cercetare și cartare a peșterii. Pentru aceasta, ei realizează întâi o schemă a legăturilor dintre culoare, apoi fac măsurătorile necesare întocmirii hărții.

Cei doi speologi plecați să înregistreze legăturile dintre culoare, procedează în felul următor:

- coboară prin culoarul vertical de la intrare pe care-l marchează (pe perete) cu numărul 1;
- ajunși în primul culoar orizontal, ei își marchează cu X un perete, față de care se orientează, apoi pornesc spre stânga; la capătul acestui culoar pot întâlni:
  - un nou puț, pe care-l marchează cu 2, îl coboară și continuă în același mod (spre stânga) investigarea sistemului de culoare accesibil prin puțul 2;
  - o fundătură, unde se opresc și își notează pe o hârtie numărul ultimului puț pe care l-au coborât, se întorc și continuă investigarea culoarului pe ramura din dreapta marcajului cu X.
- atunci când au terminat de explorat ambele ramuri ale unui culoar, ei escaladează puțul de acces, recuperează coarda cu care echipaseră puțul la coborâre și continuă explorarea culoarului superior, pe ramura din dreapta marcajului X.

În concluzie, ei explorează peștera, coborând ori de câte ori se poate coborî și virează tot timpul la stânga, astfel încât să nu treacă de două ori prin același loc, mergând în același sens.

Observații:

- Fiecare nou puț coborât se marchează cu următorul număr disponibil (nefolosit).
- Notațiile corespunzătoare fundăturilor respectă ordinea cronologică a descoperirii lor.

La întoarcere, ei fac următoarele observații:

- nu se poate ajunge în nici un coridor prin două puțuri diferite;
- două coridoare diferite nu pot comunica decât printr-un culoar vertical și nu există puț care leagă mai mult de două culoare.

Vi se cere să reconstituiți schema legăturilor subterane cu ajutorul însemnărilor celor doi.

Date de intrare:

Fișierul PESTERA.IN conține :

- pe prima linie o valoare n, reprezentând numărul de fundături întâlnite de cei doi (n≤200);
- pe linia următoare cele n numere notate de speologi pe hârtie, despărțite între ele prin câte un spațiu.

Date de ieșire:

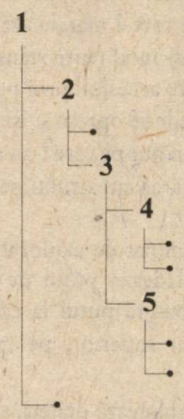
Schema legăturilor, pe care o veți reconstitui, va conține numerele de ordine ale puțurilor, pentru fiecare puț precizându-se întâi numărul puțului din stânga coridorului unde ajunge acesta, sau caracterul '*', dacă în stânga este o fundătură, apoi numărul puțului din dreapta coridorului unde ajunge acesta, sau caracterul '*', dacă în dreapta este o fundătură. Schema legăturilor (realizată ca în exemplu, folosind caracterele cu codurile 195,192,179) va fi scrisă în fișierul text PESTERA.OUT. Dacă fișierul conține asemenea numere încât acestea nu pot fi rezultatul cercetării peșterii, fișierul de ieșire va conține mesajul 'IMPOSIBIL'.

Exemple:

1) PESTERA.IN:

```
6
2 4 4 5 5 5
```

PESTERA.OUT:



2) PESTERA.IN:
5
1 2 1 3 4
PESTERA.OUT:
IMPOSIBIL

Soluția autorului:

Configurația peșterii corespunde, conform condițiilor din enunț, unui arbore binar strict. Secvența de numere notată de speologi atașează fiecărui terminal numărul de noduri interne aflate înaintea lui în parcurgerea în preordine a arborelui. Aceste secvențe asociate arborilor binari stricți poartă numele de *secvențe Pal*.

Dimensiunea vectorului este egală cu numărul de frunze ale arborelui.

Validarea valorilor din secvență se realizează fie în timpul construirii arborelui, fie pe baza observației că valorile din vectorul dat sunt crescătoare și că pe fiecare poziție i se poate afla un număr mai mare sau egal cu i.

Reconstituirea arborelui presupune construirea recursivă a legăturilor de descendență și "inventarea" unor noduri etichetate cu numere succesive până când se ajunge cu eticheta nodului curent la valoarea curentă din vectorul dat la intrare.

Procedura de afișare este implementată recursiv, variabila ind desemnând *gradul de indentare* al nodului curent, iar vectorul intors precizează despre fiecare nod i, dacă în momentul respectiv au fost "rezolvați" ambii descendenți sau a fost "tratat" numai descendentul stâng, urmând a fi "tratat" mai târziu descendentul drept.

Bibliografie:

Vajnovszki V.: *Algorithmes de generation pour certaines classes d'arbres, These de doctorat*, Université de Bourgogne, oct.1994

Listing PESTERA.PAS

```
Program Pestera;
{ Enunt si solutie: profesor Rodica Pintea          }
{ Reconstruirea unui arbore binar strict pe baza    }
{ unei codificari liniare si reprezentarea lui      }
{ in format TREE.                                   }
var a,st,dr:array[1..200]of Byte;
    intors:array[0..200]of Boolean;
    n,i,j,jj,k,term:Integer;
    ffo,ffi:Text;

procedure afisare(j:Byte; ind:Integer);
begin
  if j=0
  then
    begin Writeln(ffo,'*'); Exit end;
  Writeln(ffo,j);
  intors[ind]:=false;
  for i:=0 to ind-1 do
    if not intors[i]
    then Write(ffo,'| ')
    else Write(ffo,'  ');
  Write(ffo,'|-');
  afisare(st[j],ind+1);
  intors[ind]:=true;
  for i:=0 to ind-1 do
    if not intors[i]
    then Write(ffo,'| ')
    else Write(ffo,'  ');
  Write(ffo,'L-');
  afisare(dr[j],ind+1)
end;

procedure construieste(nod:Byte);
begin
  if nod=n then Exit;
  if nod=a[term]
  then st[nod]:=0
  else
    begin
      st[nod]:=nod+1;
      j:=j+1;
      construieste(st[nod])
    end;
  term:=term+1;
  if j=a[term]
  then dr[nod]:=0
  else
    begin
      dr[nod]:=j+1;
      j:=j+1;
      construieste(dr[nod])
    end
end;


function valid:Boolean;
```

```
begin
  valid:=false;
  if (a[1]<1) or (a[1]>n)
  then Exit;
  for i:=2 to n do
    if (a[i]<i) or (a[i]<a[i-1])
    then Exit;
  if a[n+1]<>n
  then Exit;
  valid:=true
end;

Begin
  Assign(ffi,'PESTERA.IN'); Reset(ffi);
  Assign(ffo,'PESTERA.OUT'); Rewrite(ffo);
  Readln(ffi,n);
  for i:=1 to n do Read(ffi,a[i]);
  n:=n-1;
  if not valid
  then Writeln(ffo,'IMPOSIBIL')
  else
    begin
      term:=1;
      j:=1;
      construieste(1);
      afisare(1,0)
    end;
  Close(ffi); Close(ffo)
End.
```

Clasa a XI-a: Pe litoral

Problemă propusă de prof. Daniela Lica, Liceul "I. L. Caragiale", Ploiești

Enunț:

Pe litoralul unei țări există plaje particulare, numite A și B, între care se poate circula numai cu șalupa. Pentru fiecare plajă în parte, există un orar de funcționare al șalupelor, fiecare conținând ora de plecare și ora de sosire pe cealaltă plajă. Evident, orice turist care folosește acest mijloc de transport trebuie să achite un bilet de călătorie al cărui preț poate diferi de la voiaj la voiaj.

Din păcate, proprietarii plajelor au pretenția la o taxă de ședere pe plaja lor, taxă care va fi încasată la anumite ore de la persoanele existente la acea oră pe plaja respectivă. În cazul în care o persoană sosește pe o plajă după momentul de încasare, nu plătește până la momentul de încasare imediat următor. Aceste taxe pot avea valori diferite în funcție de ora perceperii lor.

În cazul în care o șalupă pleacă sau ajunge la un anumit moment pe o plajă și exact în acel moment se percepe taxă de ședere pe acea plajă, atunci se va plăti și taxa de ședere.

Un tânăr se află pe plaja A la ora x, cunoscută. El are întâlnire cu prietena lui pe plaja B la o anumită oră y. El va încerca să-și planifice "staționările" pe plaje și "plimbările" cu șalupa astfel încât la ora y să fie la destinație, plătind taxa de ședere pe plaje și taxele de transport, iar suma totală a taxelor plătite de el să fie de valoare minimă.

Determinați orarul pe care trebuie să-l urmeze și totalul taxelor plătite.

Date de intrare:

Fișierul text `INPUT.TXT` are următoarea structură:

- pe prima linie sunt scrise două numere: `oA` ($0 \le oA \le 23$) și `mA` ($0 \le mA \le 59$) reprezentând ora inițială și minutul la care tânărul se află pe plaja A;
- pe a doua linie sunt scrise două numere: `oB` ($0 \le oB \le 23$) și `mB` ($0 \le mB \le 59$) reprezentând ora și minutul la care se vor întâlni cei doi tineri pe plaja B;
- pe a treia linie sunt scrise două numere `ntA` și `ntB` reprezentând numărul de taxe percepute pe plaja A ($0 \le ntA \le 100$), respectiv pe plaja B ($0 \le ntB \le 100$) între momentul inițial și momentul întâlnirii;
- pe următoarele `ntA` linii sunt scrise câte trei numere sub forma: `h m c` unde `h` și `m` reprezintă ora și minutele perceperii taxei pe plaja A, iar `c` reprezintă valoarea taxei ($c \le 150$);
- pe următoarele `ntB` linii sunt scrise câte trei numere sub forma: `h m c` unde `h` și `m` reprezintă ora și minutele perceperii taxei pe plaja B, iar `c` reprezintă valoarea taxei ($c \le 150$);
- următoarea linie conține numărul natural $ns \le 80$ reprezentând numărul total de șalupe care circulă între cele două plaje;
- pe următoarele `ns` linii este descris orarul șalupelor folosind câte șase numere scrise sub forma: `d hA mA hB mB c` unde:
  - `d` poate lua doar valorile `0` și `1`;
  - dacă `d` este `0`, atunci:
    - `hA` și `mA` reprezintă ora și minutele plecării șalupei din A;
    - `hB` și `mB` reprezintă ora și minutele sosirii în B;
  - dacă `d` este `1`, atunci :
    - `hA` și `mA` reprezintă ora și minutele sosirii șalupei în A;
    - `hB` și `mB` reprezintă ora și minutele plecării din B;
  - `c` reprezintă costul biletului de călătorie.

Datele sunt despărțite pe linii prin câte un spațiu, sunt corecte și admit întotdeauna soluție.

Date de ieșire:

Fișierul text `OUPUT.TXT` va avea următoarea structură:

- pe prima linie se va scrie numărul `ctm`, reprezentând costul minim al deplasării (compus din toate taxele plătite de tânăr);
- pe linia următoare se vor scrie numerele de ordine ale șalupelor cu care a călătorit tânărul, în ordinea folosirii lor.

Exemplu:

```
INPUT.TXT:
1 0
9 0
3 3
2 0 10
4 0 2
6 0 10
3 0 10
5 0 10
7 0 2
7
0 1 0 2 0 3
0 3 0 4 0 6
0 5 0 6 0 8
0 7 0 8 0 9
1 3 0 2 0 5
1 5 0 4 0 7
1 7 0 6 0 9
OUTPUT.TXT:
20
1 5 3
```

Soluția autorului:

Pentru rezolvarea problemei s-a folosit modelarea acesteia cu ajutorul unui digraf.

Acesta va avea numărul de noduri egal cu de două ori numărul șalupelor care transportă călători plus încă două noduri.

Nodul 1 reprezintă ora inițială la care se află călătorul pe plaja A iar nodul final (ntr*2+2) reprezintă ora de întâlnire de pe plaja B. Celelalte noduri au semnificația orelor de plecare sau sosire ale unei șalupe pe plaja A (începând cu nodul 2 până la 1+ntr) respectiv orele de plecare sau sosire ale șalupelor pe plaja B (de la ntr+2 până la 2*ntr+1).

Se vor trasa toate arcele existente în digraf calculând în același timp pentru fiecare dintre ele costul, ținându-se cont de taxele de pe fiecare plajă în cazul în care călătorul staționează în acea perioadă pe una din ele. Pentru arcele ce leagă nodurile dintre cele două plaje se va considera drept cost valoarea biletului.

După construirea digrafului se va calcula drumul minim de la nodul 1 la nodul 2*ntr+2.

Sursa prezentată în cele ce urmează scrie rezultatele într-un fișier al cărui nume se citește de la tastatură, iar formatul de scriere este făcut specificându-se pentru fiecare șalupă ora de plecare și ora de sosire, deci nu numărul de ordine al acestora.

Listing LITORAL.PAS

```
Program Litoral;
{ Enunt si solutie: profesor Daniela Lica }
{ Colegiul National Ion Luca Caragiale    }
type tip=array[1..100] of record
                            ora:Integer;
                            taxa:Integer
                          end;
var pz,nta,ntb,oa,ob,ntr:Integer;
    ta,tb:tip;
    f:Text;
    max:Longint;
    tr:array[1..80] of record
                         dir:Boolean;
                         a,b,cost:Integer
                       end;
    a:array[1..162,1..162] of Integer;
    tata:array[1..162] of Integer;
    cost:array[1..162] of Longint;

function valoare(cat:Integer; c:tip;
a,b:Integer):Integer;
var ret,i:Integer;
begin
  ret:=0;
  for i:=1 to cat do
    if (a<=c[i].ora) and (c[i].ora<=b)
    then ret:=c[i].taxa+ret;
  valoare:=ret
end;
procedure generare;
var i,j:Integer;
begin
  for i:=1 to ntr*2+2 do
```

Interpretarea testului:

Ora	Șalupa	Plaja inițială	Plaja finală	Taxa de ședere	Cost transport
1h 0m-2h 0m	1	A	B	-	3
2h 0m-3h 0m	5	B	A	-	5
3h 0m-5h 0m	-	A	-	2	-
5h 0m-6h 0m	3	A	B	-	8
6h 0m-9h 0m	-	B	-	2	-
				Total: 4	Total: 16

```
    for j:=1 to ntr*2+2 do a[i,j]:=-1;
  for i:=1 to ntr do
    a[1,i+1]:=valoare(nta,ta,oa,tr[i].a);
  for i:=1 to ntr-1 do
    for j:=i+1 to ntr do
      if tr[i].a>tr[j].a
      then
a[1+j,1+i]:=valoare(nta,ta,tr[j].a,tr[i].a)
      else
a[1+i,1+j]:=valoare(nta,ta,tr[i].a,tr[j].a);
  for i:=1 to ntr do
    a[ntr+i+1,2*ntr+2]:=valoare(ntb,tb,tr[i].b,ob);
  for i:=1 to ntr-1 do
    for j:=i+1 to ntr do
      if tr[i].b>tr[j].b
      then
a[ntr+1+j,ntr+1+i]:=valoare(ntb,tb,tr[j].b,tr[i].b)
      else
a[ntr+1+i,ntr+1+j]:=valoare(ntb,tb,tr[i].b,tr[j].b);
  for i:=1 to ntr do
    if tr[i].dir then a[i+1,i+ntr+1]:=tr[i].cost
                 else a[i+ntr+1,i+1]:=tr[i].cost
end;

procedure citire;
var f:Text;
    h,m,i:Integer;
begin
  Assign(f,'INPUT.TXT'); Reset(f);
  Readln(f,h,m); oa:=h*60+m;
  Readln(f,h,m); ob:=h*60+m;
  Readln(f,nta,ntb);
  for i:=1 to nta do
  begin Readln(f,h,m,ta[i].taxa);
        ta[i].ora:=h*60+m
  end;
  for i:=1 to ntb do
  begin Readln(f,h,m,tb[i].taxa);
        tb[i].ora:=h*60+m
  end;
  Readln(f,ntr);
  for i:=1 to ntr do
  begin
    Read(f,h);
    if h=0 then tr[i].dir:=true
           else tr[i].dir:=false;
    Read(f,h,m); tr[i].a:=h*60+m;
    Read(f,h,m); tr[i].b:=h*60+m;
    Readln(f,tr[i].cost)
  end;
  Close(f)
end;
```

```
procedure Dijkstra;
var vf,min,i,j:Integer;
    sel:array[1..162] of Boolean;
begin
  for i:=2 to ntr*2+2 do sel[i]:=false;
  sel[1]:=true;
  for i:=1 to ntr*2+2 do cost[i]:=a[1,i];
  for i:=1 to ntr*2+2 do tata[i]:=1;
  tata[1]:=0;
  for i:=1 to 2*ntr+1 do
  begin
    min:=-1;
    vf:=1;
    for j:=2 to 2*ntr+2 do
      if not sel[j] and (cost[j]<>-1)
         and ((cost[j]<min) or (min=-1))
      then
        begin min:=cost[j]; vf:=j end;
    sel[vf]:=true;
    for j:=1 to ntr*2+2 do
      if (a[vf,j]<>-1) and
((cost[j]>cost[vf]+a[vf,j]) or (cost[j]=-1))
      then
        begin
        cost[j]:=cost[vf]+a[vf,j];
        tata[j]:=vf
        end
  end
end;

procedure drum;
var i:Integer;
begin
  max:=cost[2*ntr+2];
  pz:=0;
  i:=2*ntr+2;
  while i<>0 do
  begin
    pz:=pz+1;
    cost[pz]:=i;
    i:=tata[i]
  end
end;

procedure ora(a:Integer);
begin
  Write(f,a div 60,'.');
  if a mod 60<10
  then Write(f,'0',a mod 60)
  else Write(f,a mod 60)
end;
```

```
procedure tiparire;
var i:Integer;
begin
  Assign(f,'OUTPUT.TXT'); Rewrite(f);
  Writeln(f,max);
  for i:=pz-1 downto 3 do
  begin
   if (2<=cost[i]) and (cost[i]<=ntr+1) and
      ((ntr+2<=cost[i-1]) and (cost[i-1]<=2*ntr+1))
   then Write(f,cost[i]-1,' ');
   if ((ntr+2<=cost[i]) and (cost[i]<=2*ntr+1)) and
      ((2<=cost[i-1]) and (cost[i-1]<=ntr+1))
   then Write(f,cost[i-1]-1,' ')
  end;
  Writeln(f); Close(f)
end;

Begin
  citire;
  generare;
  Dijkstra;
  drum;
  tiparire
End.
```

Clasa a XII-a: Alfabetizare

Problemă propusă de prof. Emanuela Mateescu, Liceul de Informatică, Iași

Enunț:

Fie A un alfabet care constă din n (1≤n≤200) simboluri, numerotate de la 1 la n. Un cuvânt peste alfabetul A este considerat corect dacă:

a) oricare două simboluri consecutive sunt distincte;

b) prin ștergerea de simboluri dintr-un cuvânt nu este posibilă obținerea unor cuvinte de forma abab (a și b simboluri distincte oarecare, din alfabetul A).

Date de intrare:

Din fișierul de intrare INPUT.TXT se citește de pe prima linie numărul n; de pe linia a doua – numărul m (1≤m≤1000) al simbolurilor din cuvânt, iar de pe următoarele m linii o succesiune de numere naturale, fiecare pe o linie separată, reprezentând simbolurile cuvântului.

Datele de intrare sunt corecte.

Date de ieșire:

Să se verifice dacă succesiunea de numere citite din fișierul de intrare reprezintă un cuvânt corect peste alfabetul A={1,2,...,n}. Să se afișeze în fișierul de ieșire OUTPUT.TXT:

- pe prima linie: mesajul 'DA', în cazul în care cuvântul citit a fost corect, respectiv 'NU' în caz contrar;
- pe a doua linie: lungimea maximă a unui cuvânt corect peste alfabetul A={1,2,...,n};
- pe următoarele linii: o succesiune de numere naturale (câte un număr pe fiecare linie), care să reprezinte un cuvânt corect de lungime maximă peste alfabetul A={1,2,...,n}.

Exemplu:

```
INPUT.TXT:
3
4
1
2
3
2
OUTPUT.TXT:
DA
5
1
2
1
3
1
```

Soluția autorului:

Spunem că o literă este de speța I dacă ea intervine o singură dată în cuvânt. În caz contrar ea se numește de speța a II-a.

Observații:

- Literele alăturate unei litere de speța a II-a trebuie să fie distincte (altfel este încălcată proprietatea b)).
- Într-un cuvânt există întotdeauna o literă de speța I (rezultă din proprietatea b)).
- Lungimea maximă a unui cuvânt care îndeplinește condițiile a) și b) este 2n-1.

Demonstrație:

Vom proceda prin inducție completă după numărul de litere din alfabet.

P(1): Lungimea maximă a unui cuvânt corect peste un alfabet cu o singură literă este 1=2*1-1 (altfel este încălcată proprietatea a)).

P(h): Lungimea maximă a unui cuvânt peste un alfabet A_k cu k litere este 2*k-1.

*P(k+1):*Lungimea maximă a unui cuvânt peste un alfabet A_{k+1} cu k+1 litere este 2*k+1.

Presupunem că *P(h)* este adevărată, pentru ∀ h≤k. Demonstrăm că *P(k+1)* este adevărată.

Fie C_{k+1} un cuvânt peste A_{k+1}. Fie a o literă de speța I din C_{k+1}. Fie b litera alăturată lui a (în stânga sau în dreapta). Dacă b este de speța I, eliminând b din cuvânt obținem un cuvânt corect peste alfabetul A_k, deci lungimea maximă a acestuia este 2*k-1. Deducem că lungimea maximă a lui C_{k+1} este 2*k-1+1<2*k+1. Dacă b este de speța a II-a, atunci fie x și y vecinii perechii ab (dacă aceștia există): xaby.

Din proprietatea b) deducem că x<>y, deci putem elimina din C_{k+1} perechea ab, fără a încălca proprietatea a).

Eliminând perechea ab din C_{k+1}, obținem un cuvânt peste Ak, lungimea maximă a acestuia fiind 2*k-1, rezultă că lungimea maximă a lui C_{k+1} este 2*k-1+2=2*k+1. Q.E.D.

Exemplu de cuvânt de lungime maximă:

1 2 3 ...n-1 n n-1 n-2 ...2 1

Listing ALFABET.PAS

```
Program alfabetizare;
{ Enunt si rezolvare: prof. Emanuela Mateescu }
const NMaxLitere=10800;
      LgMaxCuvant=2*NMaxLitere-1;
type litera=1..NMaxLitere;
     lungime=0..LgMaxCuvant;
     cuvant=array[lungime]of litera;
var n:litera;
    c:cuvant;
    lg,i:lungime;
    NrAp:array[litera]of lungime;
    eroare:Boolean;

procedure initializare;
var fin:Text;
begin
  Assign(fin,'INPUT.TXT'); Reset(fin);
  Readln(fin,n); Readln(fin,lg);
  eroare:=false;
  if lg>2*n-1
  then eroare:=true
  else
    for i:=1 to lg do
    begin
      Readln(fin,c[i]); Inc(NrAp[c[i]]);
      if c[i]=c[i-1]
      then
        begin eroare:=true; Break end
    end;
  Close(fin)
end;

procedure elimin(i:lungime);
{ procedura elimina din cuvant litera de pe pozitia i }
var j:lungime;
begin
  Dec(NrAp[c[i]]);
  for j:=i to lg-1 do c[j]:=c[j+1];
  Dec(lg)
end;

procedure afisare;
var fout:Text;
    i:litera;
begin
  Assign(fout,'OUTPUT.TXT');
  Rewrite(fout);
  if eroare then Writeln(fout,'NU')
            else Writeln(fout,'DA');
  Writeln(fout,2*n-1);
  for i:=1 to n do Writeln(fout,i);
  for i:=n-1 downto 1 do Writeln(fout,i);
  Close(fout)
end;

Begin
  initializare;
  while not eroare and (lg>1) do
  begin
    i := 1;          { caut o litera de speta I }
    while (i<=lg) and (NrAp[c[i]]<>1) do Inc(i);
    if i>lg
    then eroare:=true
    else
      begin
        Inc(i);
        while (i<=lg) and (NrAp[c[i]]=1) do
          elimin(i);
        if i>lg then i:=i-2
                else i:=i-1;
        if i>0  { mai exista o pereche in cuvant }
        { verific vecinii perechii c[i] si c[i+1] }
        then
          if (i-1>0) and (i+2<=lg)
                     and (c[i-1]=c[i+2])
          then eroare:=true
          else
            begin
              elimin(i); { elimin c[i] si c[i+1]}
              elimin(i)
            end
      end
  end;
  afisare
End.
```

Program sursă:

Puteți descărca listingurile prin Internet de la http://www.ginfo.ro/8_2/surse

Paginile „Soluții" au fost realizate de **Clara Ionescu**

E-mail: cionescu@ginfo.ro

HotSoft

KeyMap & Fonts

Add on

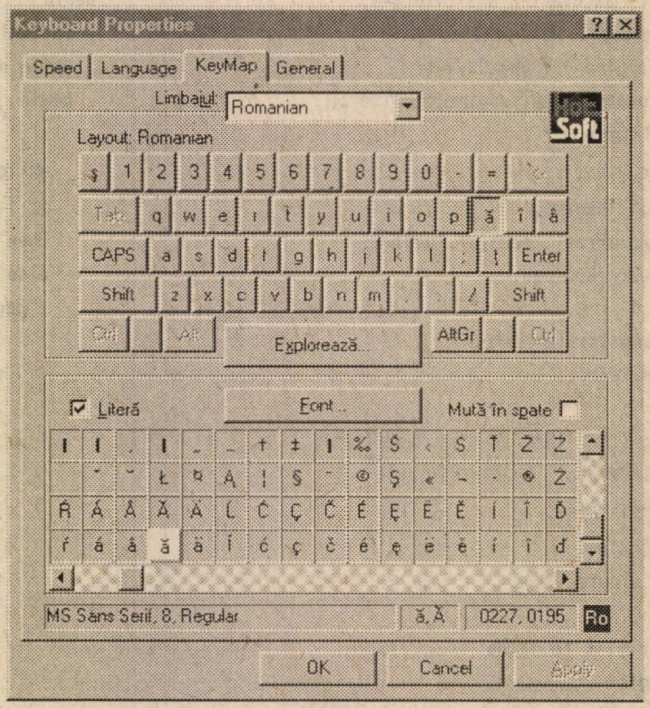
Designed for

MICROSOFT WINDOWS'95

VERSIUNEA 2

KeyMap 2.0

- este utilitarul Windows 95 de care aveți nevoie pentru a scrie comod în orice limbă.
- permite configurarea software a tastaturii în funcție de necesitățile specifice oricărui utilizator.
- pentru fiecare layout atașat unei limbi și pentru fiecare tastă a acestuia, puteți configura 4 caractere ce să fie generate de acesta!
- se instalează și se configurează foarte ușor pentru oricine a mai utilizat un produs Windows 95.
- Vă oferim de asemenea, la cerere, peste 180 de fonturi True Type sau Adobe Type (postscript), conținând toate caracterele specifice limbilor europene cu alfabet latin.



40(0)65-166516

***

www.hotsoft.ro/keymap

HotSoft SRL

Noul KeyMap 2.0 pentru Win95

împreună cu peste 180 de fonturi

Doresc să cumpăr produsul, KeyMap2.0.

V-am expediat la BCR Tg.Mureș în contul nr. 4072032198 al S.C.HotSoft SRL Tg-Mureș, 100.000 lei (în care sunt incluse și taxele poștale), cu mandatul poștal/ordinul de plată nr.: ..... din data de ....., pentru o licență a acestuia.

(Pentru licențe multiple apelați la sediul firmei.) ☐ Doresc să-mi trimiteți și factură.

Numele și prenumele: ..... Funcția: .....

Firma: .....Adresa: .....

Cod poștal/Localitate: ..... Telefon/Fax: .....

Data ..... Semnătura .....

PROBLEME propuse pentru REZOLVARE

Problemele de mai jos au fost propuse la concursul memorial „Nemes Tihamer"- 1998 - la care au participat 41 elevi din România. Credem că rezolvarea problemelor va oferi ocazia unui antrenament pentru mulți elevi pasionați de algoritmi. Mulțumim autorilor problemelor (D-l prof. Zsako Laszlo și colaboratorii săi) pentru că ne-au acordat dreptul de a le publica în Ginfo.

P029801: Ogor

Pe un teren dreptunghiular s-au semănat mai multe tipuri de plante. Despre fiecare metru pătrat al terenului se știe cu ce fel de plantă a fost semănat. Se numește *tabelă de plantă* acel dreptunghi de arie maximă pe care s-a semănat un același tip de plantă.

În prima linie a fișierului `PLANTE.IN` sunt scrise numerele `M` și `N`, reprezentând dimensiunile terenului ($1 \le N \le 50$ și $1 \le M \le 50$); numerele sunt despărțite printr-un spațiu. În următoarele `MxN` linii ale fișierului sunt scrise denumirile plantelor cu care s-a semănat fiecare metru pătrat. Matricea se traversează linie după linie.

Să se scrie un program care afișează pe ecran și scrie în fișierul `TEREN.IN` numărul tabelelor de plante.

Exemplu:

```
TEREN.IN:
     4 5
     Grau
     Grau
     Grau
     Cartof
     Cartof
     Grau
     Grau
     Grau
     Cartof
     Cartof
     Grau
     Grau
     Porumb
     Porumb
     Porumb
     Grau
     Grau
     Porumb
     Porumb
     Porumb
TEREN.OUT:
     4
```

P029802: Grăniceri

În Țara lui Papură Vodă locuitorii sunt identificați pe baza numelui. Dintre locuitori cel mult 500 au pașaport. Țara are `N` (<=9) puncte de trecere a frontierei. Se ține evidența intrărilor și ieșirilor.

Evidențele anului precedent sunt în fișierul `VAMA.INy` în ordinea trecerii frontierei de către călători. y reprezintă numărul de ordine al punctului de frontieră. Fiecare linie conține numărul de ordine al zilei din an în care a avut loc trecerea frontierei, cuvântul `IN` sau `OU` (corespunzător intrării, respectiv ieșirii în/din țară) și numele persoanei (un nume este un șir de caractere de lungime <=40).

Să se scrie un program care:

A. scrie în fișierul `AFARA.OUT` numele persoanelor care în ultima zi a anului sunt în străinătate.

B. scrie în fișierul `FRAUDE.OUT` numele acelor persoane care au încălcat regula stabilită: nu se poate intra sau ieși consecutiv de mai multe ori în/din țară.

Exemplu:

```
VAMA.IN1:
22 OU Pop Vasile
23 IN Pop Vasile
36 IN Rus Maria
44 OU Cuc Ion
170 IN Ion Ion
192 IN Pop Vasile
```

```
VAMA.IN2:
38 OU Rus Maria
167 OU Ion Ion
178 OU Ion Ion
188 OU Cuc Ion

AFARA.OUT:
Rus Maria
Cuc Ion
Ion Ion

FRAUDE.OUT:
Cuc Ion
Pop Vasile
```

P029803: Camping

În campingul „Făget" există K căsuțe care pot fi închiriate 365 zile din an. Orice turist poate închiria o căsuța din camping exact M zile din an. Campingul a adunat solicitările pe un an întreg, cu un an înainte. Fiecare turist precizează prima zi începând cu care dorește să închirieze (pe o perioadă de M zile) o căsuță.

Să se scrie un program care determină numărul maxim de turiști care vor putea fi primiți în camping.

În prima linie a fișierului CAMP.IN este scris numărul K (1<=K<=100) de căsuțe din camping. În a doua linie este scris numărul M (1<=M<=365). În a treia linie este scris numărul solicitărilor N (1<=N<=1000). În următoarele N linii sunt scrise solicitările. Rezultatul se va scrie în fișierul CAMP.OUT și se va afișa pe ecran.

Exemplu:

```
CAMP.IN:
2
7
8
1
10
2
11
1
3
4
18

CAMP.OUT:
5
```

P 029804: Expediție pe Marte

Pe suprafața planetei Marte se adună mostre de rocă cu ajutorul unor vehicule speciale. Aceste vehicule se deplasează conform unei scheme de caroiaj doar spre sud sau est și doar pe un teren lipsit de stânci. Harta pusă la dispoziție conține informații cu privire la pozițiile unde se află roci (o rocă într-o poziție) și unde sunt stânci. Harta este dată sub forma unei matrice de dimensiune MxN care conține elemente egale cu 0, 1 și 2, având semnificația:

- 0 înseamnă teren gol
- 1 înseamnă rocă
- 2 înseamnă stâncă

Vehiculele pornesc din poziția (1,1) care este colțul din stânga sus a hărții. Locul unde trebuie să ajungă vehiculele este în colțul din dreapta jos al hărții (poziția (M,N)).

Să se scrie un program care determină numărul maxim de roci posibil de adunat, presupunând că avem un număr suficient de mare de vehicule.

Prima linie a fișierului MARTE.IN conține numerele M și N, despărțite printr-un spațiu (1<=M, N<=100). În următoarele M linii vom avea elementele liniilor matricei corespunzătoare hărții, despărțite prin câte un spațiu.

Să se scrie numărul maxim de roci pe ecran și în fișierul MARTE.OUT.

Exemplu:

```
MARTE.IN:
    10 12
    0 1 2 1 0 0 1 0 2 0 0 0
    0 1 0 0 0 2 0 0 0 0 0 0
    0 0 0 1 0 0 0 0 2 0 0 0
    2 0 0 0 0 2 0 0 0 0 0 0
    0 0 0 1 0 0 0 0 0 0 0 0
    0 2 0 0 2 0 0 0 0 0 0 0
    1 0 0 0 0 1 1 1 1 0 0 0
    1 1 1 1 1 2 0 0 0 1 0 0
    1 0 0 0 0 0 0 0 0 1 0 0
    1 0 0 0 0 0 0 0 0 1 0 0

MARTE.OUT:
    14
```

P029805: Hotel

Pentru un lanț de hoteluri se ține evidența bucătarilor și cofetarilor. În această evidență se pot realiza căutări după nivelul profesional și după vârstă.

Fișierele text BUCATAR.IN și COFETAR.IN conțin informații despărțite printr-un spațiu, structurate pe linii (o linie pentru o persoană):

- nivelul profesional (1 corespunde celui mai ridicat nivel, 10 celei mai slabe pregătiri),
- vârsta (număr întreg cuprins între 15 și 100),
- numele (format din cel mult 40 caractere litere din alfabetul englez, cifre și spații).

În fiecare din cele două fișiere sunt informații despre cel mult 100 de persoane.

Întrebările la care trebuie să răspundă la un moment dat programul sunt în fișierul INTREB.IN sub forma: xxxx:yyyy:zzzz, unde câmpurile sunt despărțite prin caracterul : (două puncte), iar xxxx reprezintă profesia, yyyy reprezintă nivelul profesional și zzzz vârsta persoanei (persoanelor) căutate.

xxxx = BUCATAR sau COFETAR sau ORICE
ORICE are semnificația că persoana poate fi și bucătar, și cofetar.
yyyy =<N sau >N sau =N sau <=N sau >=N
yyyy precizează relația dintre nivelul profesional și numărul întreg N.
zzzz =MIN sau MAX
zzzz precizează informația cu privire la preferința prioritară pentru cea mai tânără sau cea mai în vârstă persoană.

Numele persoanelor găsite corespunzător criteriilor conținute în fișierul INTREB.IN se vor scrie în fișierul RASPUNS.OUT, un nume pe o linie. Dacă nu există nici o persoană care să corespundă criteriilor, în linia respectivă se va scrie 'NU ESTE'. Dacă sunt mai multe persoane se va alege prima persoană în ordine alfabetică după nume.

Exemplu:

```
BUCATAR.IN:
      1 35 Pop Vasile
      4 49 Muresan Ion

BUCATAR.IN:
      1 35 Pop Vasile
      4 49 Muresan Ion

INTREB.IN:
      BUCATAR:=4:MAX
      ORICE:<2:MAX
      COFETAR:>=3:MIN

RASPUNS.OUT:
      Muresan Ion
      Rus Maria
      NU ESTE
```

P029806: Radar

O bază aeriană este înzestrată cu un radar special care supraveghează spațiul aerian deasupra unei suprafețe de dimensiune 1000x1000. Radarul are rolul de a sesiza apariția unei rachete inamice. Colțul din stânga sus a suprafeței are coordonatele (1,1).

Baza este plasată pe un pătrat aflat între punctele de coordonate (496,496) și (505,505). Într-o unitate de timp radarul „vede" o singură porțiune de dimensiune 1000x100. (Începe cu prima porțiune, apoi continuă cu a doua ș.a.m.d., apoi reia din nou cu prima.)

Porțiune 1	Porțiune 2	...	Porțiune 10

Radarul poate sesiza o rachetă doar dacă aceasta se află în porțiunea supravegheată în unitatea de timp curentă.

Rachetele sunt lansate din afara suprafeței supravegheate. Fiecare rachetă traversează un drum rectiliniu, cu viteză constantă, astfel încât ambele coordonate care îi precizează poziția se modifică de la o valoare întreagă la alta, diferența dintre ele fiind egală cu cel mult 10. La un moment dat, în aceeași poziție se pot afla mai multe rachete. *Racheta lovește solul dacă la începutul unității de timp se află deasupra bazei. Racheta poate să fie distrusă în timpul zborului* (de exemplu, poate fi anihilată).

Soldații din bază trebuie alertați dacă o rachetă se apropie de bază și dacă această rachetă a fost sesizată de trei ori de radar. Apropierea unei astfel de rachete trebuie semnalizată o singură dată.

Să se scrie un program care, în caz de nevoie, alertează soldații.

Datele de intrare se află în fișierul RACHETA.IN. Fișierul are cel mult 1000 linii și în fiecare linie se află trei date (numere întregi, despărțite de câte un blanc) referitoare la o sesizare: unitatea de timp în care a avut loc sesizarea, linia și coloana care determină poziția rachetei în momentul sesizării.

Datele de ieșire se vor afișa pe ecran și în fișierul de ieșire RACHETA.OUT și constă din datele de semnalizare, adică alertarea. În fiecare linie se vor scrie 4 numere întregi despărțite de câte un blanc: unitatea de timp în care are loc alertarea, unitatea de timp la care se așteaptă lovirea solului, linia și coloana care precizează poziția în care racheta va lovi solul.

Exemplu:

```
RACHETA.IN:
      1 1 1
      7 506 5
      11 21 21
      17 526 35
      21 41 41
      27 546 65
      31 61 61
      37 566 95
      41 81 81
      48 588 128
      51 101 101
      61 121 121

RACHETA.OUT:
      21 249 497 497
```

P029807: Închirieri auto

O întreprindere de transporturi închiriază autovehicule. Unul din vehicule este extrem de solicitat, de aceea, solicitările pentru anul următor se adună pe parcursul anului curent. Solicitarea se precizează printr-o pereche de numere A B, unde A reprezintă numărul de ordine al zilei din an începând cu care clientul dorește să închirieze vehiculul și B reprezintă numărul de ordine al zilei din an, când închirierea încetează. Întreprinderea vrea să planifice închirierile cât mai profitabil și în acest scop acceptă doar acele solicitări care, satisfăcute integral, asigură (în totali-

tatea lor) o închiriere pe un număr maxim posibil de zile.

Să se scrie un program care determină, pe baza solicitărilor, numărul maxim posibil de zile în care vehiculul poate fi închiriat.

Fișierul AUTO.IN conține solicitările. Prima linie conține numărul de solicitări N (1≤N≤1000). În următoarele N linii sunt solicitările, adică perechi de numere A B, despărțite de câte-un spațiu (1≤A≤B≤365).

Rezultatul se va scrie pe ecran și în fișierul AUTO.OUT.

Exemplu:

```
AUTO.IN:
      10
      1 23
      6 37
      24 43
      46 71
      12 54
      44 66
      13 25
      22 33
      5 10

AUTO.OUT:
      69
```

P029808: Robot

Într-o fabrică, s-a automatizat procesul de producție. Mașinile sunt așezate într-o hală conform unui caroiaj. La sfârșitul zilei de lucru, piesele fabricate de mașini sunt adunate de roboți. Roboții se deplasează pe pistele lor în spațiul de deasupra mașinilor, corespunzător aceluiași caroiaj. Roboții pornesc din colțul din stânga sus (1,1) și trebuie să ducă piesele în colțul din dreapta jos. Roboții sunt programați astfel încât se deplasează doar în dreapta și în „jos".

Să se scrie un program care determină numărul minim de roboți care trebuie porniți pentru a se strânge toate piesele fabricate, știind că un robot poate transporta un număr oarecare de piese.

Harta halei se dă sub forma unui tablou bidimensional de dimensiune MxN, având elemente egale cu 0 și 1. Valoarea 1 are reprezintă o mașină, valoarea 0 reprezintă spațiu liber în hală.

Fișierul ROBOT.IN conține în prima linie valorile M și N, despărțite de câte un spațiu (1≤M,N≤100). Pe următoarele M linii sunt elementele tabloului corespunzător halei. Valorile 0 și 1 sunt despărțite de câte un spațiu.

Rezultatul se va scrie în fișierul ROBOT.OUT.

Exemplu:

```
ROBOT.IN:
      10 12
      0 1 1 1 0 0 1 0 1 0 0 0
      0 1 0 0 0 1 0 0 0 0 0 0
      0 0 0 1 0 0 0 0 1 0 0 0
      1 0 0 0 0 1 0 0 0 0 0 0
      0 0 0 1 0 0 0 0 0 0 0 0
      0 1 0 0 1 0 0 0 0 0 0 0
      1 0 0 0 0 1 1 1 1 0 0 0
      1 1 1 1 1 1 0 0 0 1 0 0
      1 0 0 0 0 0 0 0 0 1 0 0
      1 0 0 0 0 0 0 0 0 1 0 0

ROBOT.OUT:
      5
```

P029809: Compilator

Se consideră un limbaj de programare. Un program scris în acest limbaj poate încărca cu date o porțiune de memorie având 1024 de locații de dimensiune 8 biți (începând cu prima locație).

Instrucțiunile limbajului:

`Eticheta: x(BUC)` scrie valoarea lui x în BUC locații (0≤x, BUC≤255)

`Eticheta: 'x'(BUC)` scrie codul caracterului x în BUC locații (0≤BUC≤255)

`Eticheta: 'xxxxxx'` scrie codurile caracterelor șirului de caractere xxxxxx în următoarele locații ale memoriei

`Eticheta: <X>` scrie adresa X (16 biți) în următoarele locații de memorie (prima locație conține octetul inferior, a doua octetul superior)

`Eticheta: [X]` scrie conținutul adresei X (8 biți) în următoarea locație

`Eticheta:` este un șir de caractere format din maxim 6 litere ale alfabetului englez, după care este scris caracterul „două puncte". Prezența etichetei este opțională. Pot exista oricâte blancuri, dar nu in interiorul numerelor și etichetelor.

Să se scrie un algoritm prin care un program (scris corect) se citește din fișierul PROGRAM.IN. Acest program trebuie „compilat". Să se afișeze pe ecran și să se scrie în fișierul PROGRAM.OUT rezultatul „compilării" (conținutul locațiilor de memorie).

În fișierul PROGRAM.IN fiecare instrucțiune este scrisă pe linie nouă.

Exemplu:

```
PROGRAM.IN:
      A: 12 (2)
         'a'(3)
      B: "bcd"
         [A]
         < A >
         <C>
      C: [B]

PROGRAM.OUT:
      12 12 97 97 97 98 99 100 12 1 0 14 0 98
```

Așteptăm soluții pe adresa redacției (preferabil prin e-mail), până la data de 1 mai 1998.

Algoritmul BOYER-MOORE

de Cosmin Truță

Odată cu spectaculoasa creștere a puterii microprocesoarelor, problema performanței programelor pare să se rezolve de la sine. Din păcate, nu este chiar așa. Adesea, impactul utilizării unui algoritm mai eficient asupra performanței de ansamblu a unei aplicații poate să fie foarte important, mai ales dacă este vorba de o operație care se execută în mod frecvent.

De multe ori, în conceperea unor algoritmi sau în dezvoltarea unor aplicații complexe, apare necesitatea de a căuta aparițiile unui șir de caractere într-un alt șir, de obicei, mai lung. Poate să apară, de exemplu, o problemă care să implice căutarea aparițiilor unui text într-o categorie de fișiere, mai mult sau mai puțin voluminoase.

O soluție imediată ar fi următoarea: citim fiecare fișier linie cu linie și verificăm dacă textul este cuprins în acea linie. Adică, atunci când primul caracter din text îl găsim într-o poziție oarecare din linie iar următoarele caracterele din text se află în poziții consecutive în linia din fișier, înseamnă că am găsit o apariție a textului. Probabil nu puțini dintre cititori cunosc funcția `Pos` din biblioteca Turbo Pascal, sau funcția `Strstr` din biblioteca standard a limbajului C.

Un important neajuns al acestei metode este ilustrat în exemplul următor: să presupunem că dorim să căutăm subșirul `„ALA!"` în șirul `„ALA-BALA-PORTOCALA!"`. Vom găsi caracterul `'A'` chiar pe prima poziție a șirului. Comparând următoarele caractere, observăm: `'L'='L'`, `'A'='A'`, `'-'≠'!'`. Următoarea poziție pe care regăsim caracterul `'A'` este poziția a treia, dar observăm că `'-' ≠ 'L'`. Continuăm cu `'A'` pe poziția a șasea: `'L'='L'`, `'A'='A'`, `'-'≠'!'`, și nu vom avea mai multe șanse nici cu poziția a opta. În final, vom găsi subșirul chiar la sfârșitul șirului: `'A'='A'`, `'L'='L'`, `'A'='A'`, `'!'='!'`.

Ne punem problema dacă putem elimina, pe cât posibil, o parte din suita de comparații. În treacăt, pot aminti că o soluție notabilă a fost oferită de *Knuth*, *Morris* și *Pratt* în 1977. Dar cel mai rapid algoritm de căutare cunoscut în prezent, un algoritm de o extraordinară eleganță, este algoritmul **Boyer-Moore**, care datează din același an, și care va fi prezentat în articolul de față.

Ideea

Să luăm același șir, `„ALA-BALA-PORTOCALA!"`, în care vom căuta apariția subșirului `„ALA!"`. Dacă privim mai atent, observăm caracterul `'-'` de pe poziția a patra din șir, deja suntem siguri că pe primele patru poziții nu vom găsi sub nici o formă subșirul căutat. De ce? Pentru că subșirul, cu o lungime de patru caractere, nu conține deloc caracterul `'-'`, și nu mai are rost să verificăm nici celelalte trei caractere, de pe primele trei poziții:

```
ALA-BALA-PORTOCALA!
ALA!
 *ALA!
 **ALA!
 ***ALA!
```

Generalizând, vom compara șirurile caracter cu caracter, dar, de data aceasta, vom porni dinspre sfârșitul subșirului, și nu de la început. În acest mod, putem „sări" peste un număr însemnat de comparații inutile. Chiar în exemplul dat, după prima comparație (ultimul element din subșir, cu elementul corespunzător din șir), am eliminat deja alte trei comparații care nu ar fi dus dealtfel la nici un rezultat.

Ne rămâne deci să căutăm în șirul `„...BALA-PORTOCALA!"`. Analizăm din nou al patrulea caracter din șir - de data aceasta, caracterul `'A'`. Față de situația precedentă, caracterul `'A'` apare de două ori în subșir, pe prima poziție, respectiv pe a treia poziție. Aceasta înseamnă că ar fi posibil ca apariția subșirului în șir să aibă una din cele două forme:

```
ALA-BALA-PORTOCALA!
*****ALA!
```

sau,

```
ALA-BALA-PORTOCALA!
*******ALA!
```

Pentru a fi siguri că nu vom pierde nici o eventuală apariție a subșirului, vom lua în considerare prima variantă, unde saltul este mai mic. Să nu uităm: tendința este ca salturile să fie mari, dar fără a periclita siguranța căutării. Observăm și aici că primul caracter, `'B'`, este ignorat în ambele situații.

Avem subșirul `"ALA!"` care trebuie căutat în subșirul „ALA-PORTOCALA!". Vom repeta raționamentul, trecând, succesiv, prin următoarele stări:

a: `ALA-BALA-PORTOCALA!`
`*****ALA!`

Nu există caracterul `'-'` în subșir, deci avansăm cu un număr de caractere egal cu lungimea subșirului.

b: `ALA-BALA-PORTOCALA!`
`*********ALA!`

Nu există caracterul `'T'` în subșir, deci avansăm din nou, cu același număr de caractere.

c: `ALA-BALA-PORTOCALA!`
`**************ALA!`

Caracterul `'L'` se găsește în subșir pe poziția a doua. Avansăm astfel încât caracterul din șir să se suprapună cu cel din subșir.

d: `ALA-BALA-PORTOCALA!`
`****************ALA!`

După care, reluând operațiile, dinspre sfârșit înspre început, aflăm de această dată că ne găsim chiar pe poziția corectă.

Câteva detalii

Pentru a codifica într-un algoritm pașii descriși anterior, sunt necesare o serie de pregătiri prealabile. Ele constau într-o scurtă analiză a subșirului căutat relativ la mulțimea caracterelor posibile care pot apare în șir. În urma analizei va rezulta o tabelă ce va conține, în dreptul fiecărui caracter, numărul maxim de poziții cu care poate avansa căutarea. Valorile cuprinse în această tabelă vor fi egale cu lungimea subșirului pentru caracterele ce nu apar în subșir, respectiv cu minimul distanței de la caracter la ultimul caracter din subșir, în caz contrar. Este clar că singurul caracter căruia îi corespunde valoarea zero este ultimul caracter din subșir, și nu există nici un caracter căruia să îi corespundă o valoare mai mare decât lungimea șirului.

Fie `pattern` subșirul pe care îl căutăm, `text` șirul în care se face căutarea, iar `delta`, tabela despre care am vorbit mai înainte.

Considerăm

```
PL=Length(pattern),
TL=Length (text).
```

Normal, secvența de căutare ar arăta cam așa:

```
k:=PL;
while (k ≤ TL) and (delta[text[k]] > 0) do
  k:=k+delta[text[k]];
{ ... }
```

Deoarece cea mai mare parte din timpul afectat căutării se va consuma în această buclă, ne punem problema de a o optimiza cât mai mult. La fiecare ciclare se efectuează două comparații. Un simplu artificiu ne permite însă să înglobăm cele două comparații într-una singură.

Este vorba despre faptul că nu vom avea `delta[pattern[PL]]=0`, ci vom înlocui valoarea zero cu o valoare suficient de mare, astfel încât, după ce se va incrementa `k` cu valoarea respectivă, va avea loc relația `k > TL`, adică se va ieși din buclă. În plus, va trebui să se deosebească situația epuizării textului de situația întâlnirii unei astfel de valori. Fie `large` această valoare, iar condiția necesară și suficientă căutată este `large ≥ TL + PL`.

Algoritmul

Pentru descrierea algoritmului, am ales o variantă structurată, propusă de *R.N.Horspool*, care a adus o contribuție algoritmului original și prin faptul că a definit puțin altfel valoarea de salt atunci când, în urma comparării caracter cu caracter între `pattern` și `text`, se constată diferențe. Această valoare, `cshift` (inițial, era un set de valori), va fi egală cu distanța cea mai scurtă între ultimul caracter din `pattern` și un alt caracter din `pattern` de aceeași valoare, dacă un astfel de caracter există. Adică:

```
cshift:=min(PL - j)
```

unde `(1 ≤ j < PL, pattern[j] = pattern[PL])`.

`cshift` va avea valoarea `PL`, dacă ultimul caracter din `pattern` nu mai apare altundeva în `pattern`.

Algoritmul prezentat va căuta și va afișa toate pozițiile aparițiilor subșirului `pattern` în șirul `text`.

În loc de încheiere

Nu cred că este nevoie să subliniez faptul că algoritmul se poate aplica și când nu este vorba doar de șiruri de caractere. Dar doresc să atrag atenția că algoritmul este aplicabil când mulțimea de valori a elementelor șirului este finită, și este eficient când această mulțime are un număr relativ redus de elemente. Setul de caractere ASCII este un bun exemplu al aplicabilității algoritmului, dar, dacă se știe că avem caractere cuprinse într-un domeniu mai restrâns (doar cifre, litere, litere mari etc.) atunci situația se poate exploata și mai eficient. În schimb, când vorbim despre șiruri de numere reale, sau alte mulțimi cu domeniu infinit sau foarte mare, sunt necesare câteva modificări, impuse de faptul că informațiile conținute de vectorul `DELTA` nu se mai pot stoca în același fel. Algoritmul se poate rescrie, dar eficiența sa nu va mai fi aceeași.

Important este faptul că, pentru un șir de lungime `m` și un subșir de lungime `n`, în cazul cel mai nefavorabil, o căutare simplă va dura un timp proporțional cu `m×n`, în timp ce, tot în cazul cel mai nefavorabil, o căutare Boyer-Moore se va executa într-un timp proporțional cu `m+n`. Și nu numai atât: în foarte multe situații o căutare BM va fi cu atât mai rapidă, cu cât subșirul căutat este mai lung, ceea ce nu se poate spune despre ceilalți algoritmi. Iar pentru căutări multiple ale unui subșir, este suficient ca secvența de inițializare (marcată cu un comentariu în algoritm) să se execute o singură dată.

O îndrăzneață aplicabilitate a fost sugerată de nume-

Algoritmul Boyer-Moore

```
Algoritmul Boyer-Moore:
  citește pattern,text
  PL ← Length(pattern)                          { initializari }
  TL ← Length(text)
  large ← PL+TL
  for c ← first_character,last_character do
    delta[c] ← PL
  for i ← 1,PL-1 do
    delta[pattern[i]] ← PL-i
  cshift ← delta[pattern[PL]]
  delta[pattern[PL]] ← large
  while k ≤ TL do                               { cautarea }
    while k ≤ TL do
      k ← k+delta[text[k]]
    if k > large
      then                                      { mai exista caractere }
        k ← k-large
        i ← k-1
        j ← PL-1
        while (j > 0) and (text[i] = pattern [j]) do
          i ← i-1
          j ← j-1
        if j = 0
          then                                  { am gasit subsirul }
            scrie i+1
            k ← k+PL
          else                                  { nu am gasit subsirul }
            k ← k+cshift
```

roși cercetători în căutarea subsecvențelor genetice din lanțul moleculei de ADN. Eu doar vă sugerez să implementați algoritmul în limbajul preferat și să-i verificați comportarea comparativ cu funcțiile echivalente oferite de bibliotecă. Iar „vrăjitorilor" dornici de performanță, le voi arunca chiar mănușa: o funcție de căutare, scrisă în limbaj de asamblare, optimizat, pe 32 de biți. Ce ziceți?

Cosmin Truță este student, anul IV, la Facultatea de Matematică, Secția Informatică, Universitatea "Babeș-Bolyai", Cluj.

Bibliografie:

1. Boyer, R.S. and Moore, J.S., *A fast string searching algrithm*, Comm. ACM 20, 10 (1977), pag. 762-772
2. Sunday, D.M., *A very fast substring search algorithm*, Comm. ACM 33, 8 (Aug. 1990), pag. 132-142
3. ***, *Notes on a very fast substring search algorithm*, Comm. ACM 35, 4 (Apr. 1992), pag. 132-137

DURATA de execuție a ALGORITMILOR

de Raluca Vernic

Se spune, printre programatori, că o problemă se poate rezolva oricât de ineficient, dar totuși corect. Ba mai mult, teoria calculabilității a găsit și o demonstrație formală pentru ceea ce părea o butadă. Dar, în lumea reală, o rezolvare ineficientă adesea nu este o rezolvare, ci o nouă problemă.

Performanțele calculatoarelor de azi ne-au obișnuit să programăm fără a ține cont prea mult de numărul de calcule elementare necesare pentru execuția unui algoritm. Totuși, anumite probleme pentru care s-au scris algoritmi corecți, rămân de nerezolvat deoarece execuția algoritmilor respectivi ar necesita ani, chiar și pe calculatoarele cele mai rapide. Din acest punct de vedere, problemele se împart în două categorii:

a) Probleme ce necesită un timp de calcul foarte mare pentru orice algoritm utilizat. O asemenea problemă este cea a comis-voiajorului care trebuie să traverseze un număr mare de orașe, să presupunem, mai multe decât 200. Se pune problema determinării unui drum de lungime minimă, care să treacă cel puțin o dată prin fiecare oraș. Orice algoritm de rezolvare a acestei probleme necesită enumerarea tuturor cazurilor posibile, iar durata de execuție crește cu $n!$, n fiind numărul de orașe. Doar aproximările euristice permit găsirea rapidă a unei soluții, fără a garanta însă că aceasta este optimă.

b) *Probleme care necesită o durată de calcul minimă sau enormă, în funcție de algoritmul de rezolvare utilizat.* De exemplu, fiind dat un număr întreg k, foarte mare, ($k>10^9$), să se afle dacă acest număr este prim sau nu. Dacă algoritmul se bazează pe căutarea secvențială a unui divizor mai mic decât k, atunci timpul de execuție va fi foarte mare. Dar acest algoritm poate fi optimizat, în consecință timpul de execuție poate fi redus simțitor, ținând cont de proprietățile numerelor prime.

Instrumente

În oricare dintre cele două cazuri anterioare s-ar încadra problema de rezolvat, se observă că este necesar ca pentru un algoritm dat să descoperim "instrumente" pentru:

1) evaluarea *ordinului de mărime* al duratei de execuție (complexitate), independent de calculatorul utilizat;

2) compararea timpului de execuție a doi algoritmi în ideea de a stabili pe cel care este mai rapid pentru a rezolva problema considerată într-un context dat;

3) stabilirea faptului dacă un algoritm dat este "optimal" sau nu, adică dacă este necesar sau nu să căutăm să-i îmbunătățim durata de execuție.

Pentru a satisface aceste trei condiții, trebuie să ajungem la o definiție a *ordinului de mărime* al duratei de execuție a unui algoritm, definiție care să nu depindă de calculatorul utilizat. În acest scop, vom modela un calculator după cum urmează:

♦ un calculator conține o *memorie* împărțită în *cuvinte* (*celule*); fiecare cuvânt conține o informație elementară, de exemplu, un număr întreg.

♦ neglijând durata de acces la instrucțiuni, *durata de execuție* va fi alcătuită din:

- durata de acces la cuvinte;
- durata de calcul propriu-zis (comparații, operații aritmetice etc.)

Exemplul 1:

Să presupunem că avem de executat instrucțiunea A←B+C. Să se evalueze durata de execuție.

Soluție:

Durata de execuție cuprinde:

♦ accesul la celula lui B (citirea);

♦ accesul la celula lui C (citirea);

♦ adunarea lui B la C;

♦ accesul la celula lui A (scrierea).

Fie λ durata necesară pentru o citire sau scriere și μ pentru o adunare. Durata de execuție pentru instrucțiune este deci: $t = 3\lambda + \mu$.

În cazul general (pentru simplificare, presupunem că toate operațiile aritmetice au aceeași durată, precum și operațiile de scriere/citire), dacă execuția unui algoritm presupune:

a — accese la memorie,
b — operații aritmetice și
c — comparații,

atunci durata de execuție este $t = \lambda a + \mu b + \nu c$, unde λ, μ și ν sunt, respectiv, durata de acces la memorie, durata unei operații aritmetice și a unei comparații.

Pentru un anumit calculator, se poate scrie deci

$k_1(a+b+c) < t < k_2(a+b+c)$

unde:

$$\begin{cases} k_1 = \inf(\lambda,\mu,\nu) \\ k_2 = \sup(\lambda,\mu,\nu) \end{cases}$$

Definiția 1:

Spunem că $(a+b+c)$ este ***ordinul de mărime*** al duratei de execuție a algoritmului considerat (***complexitatea algoritmului***), ordin de mărime care este evident independent de calculatorul ales. Vom nota $t = O(a+b+c)$ pentru a exprima faptul că durata de execuție este echivalentă cu $(a+b+c)$ până la o constantă multiplicativă.

Definiția 2:

Spunem că $f(n)=O(g(n))$ dacă

$$\exists n_0, \exists M > 0, \forall n \geq n_0, \left|\frac{f(n)}{g(n)}\right| < M.$$

Vom scrie astfel că $t=O(n^k)$ pentru cazul în care t/n^k este majorat de o constantă când $n \to \infty$

Ordinele de mărime cele mai frecvent întâlnite sunt, în funcție de parametrul *n*:

$\log n$, $n\log n$, n^2, n^3, n^4, 2^n, $n!$, n^n.

Exemplul 2:

Fie un algoritm care depinde de un parametru întreg *n* și se execută într-o oră pentru $n = 100$ pe un anumit calculator, într-un anumit limbaj. Cât timp va necesita execuția sa pentru $n = 1000$, dacă ***ordinul de mărime*** al duratei sale de execuție (complexitatea) este $\log n$, n, $n\log n$, n^2 ?

Soluție:

ordin de mărime	durata
$\log n$	1h 30'
n	10h
$n\log n$	15h
n^2	100h

Complexitatea unor algoritmi simpli

Exemplul 3: Suma a *n* numere

Avem de calculat suma a *n* numere, $a_1,\ldots,a_n$. Algoritmul utilizat se rezumă la instrucțiunea $S \leftarrow a_1 + \ldots + a_n$. Care este ordinul de mărime a duratei de execuție (complexitatea)?

Soluție:

Se observă că sunt necesare $n + 1$ accesări ale memoriei și $n - 1$ adunări. Ordinul de mărime este deci: $n + 1 + n-1=2n$. Făcând abstracție de constanta multiplicativă, durata de execuție a acestui algoritm este de ordinul *n* și se exprimă astfel: $t = O(n)$.

Exemplul 4: Produsul a două matrice

Să considerăm două matrice pătratice, `A` și `B` având `n` linii și `n` coloane. Este cunoscut algoritmul care calculează produsul `C=A×B`, conform celei mai naturale metode, dedusă din definiția produsului:

```
for i:=1 to n do
  for j:=1 to n do
begin
  S←0;
 for k:=1 to n do S←S+A[i,k]×B[k,j];
 C[i,j]←S;
end;
```

Care este ordinul de mărime al duratei de execuție a acestui algoritm?

Soluție:

Ciclul **for** exterior se execută de `n` ori. Ciclul controlat de `j` se execută de `n` ori, iar cel controlat de `k` tot de `n` ori. Avem deci:

o execuție `i←1; n-1` execuții pentru `i←i+1`
`n` execuții de tipul `"i egal n?"`
`n` execuții `j←1; n(n-1)` execuții pentru `j←j+1`
n^2 execuții de tipul `"j egal n?"`
n^2 execuții `S←0`
n^2 execuții `k←1;` n^2`(n-1)` execuții pentru `k←k+1`
n^3 execuții de tipul `"k egal n?"`
n^3 execuții `S←S+A[i,k] × B[k,j]` și n^2 execuții `C[i,j]←S`.

În ipoteza că o întrebare de genul `"i egal n?"` necesită două accesări ale memoriei plus o comparație și că accesul la o adresă ca `A[i,k]` necesită trei accesări ale memoriei plus o adunare, durata de execuție va fi proporțională cu:

$2+4(n-1)+3n+$	i
$+2n+4n(n-1)+3n^2+$	j
$+2n^2+$	S←0
$+2n^2+4n^2(n-1)+3n^3+$	k
$+12n^3+$	pentru S←S+...
$+5n^2$	pentru C[i,j], adică $t = O(n^3)$.

Deci, într-o primă aproximare, dacă produsul a două matrice de dimensiuni 10×10 necesită o secundă pe un anumit calculator, atunci produsul a două matrice de 100×100 va necesita aproape două minute pe același calculator.

Exemplul 5: Algoritmul lui Strassen

În 1969, Strassen a descoperit că produsul a două matrice pătratice de ordin 2 se poate efectua cu ajutorul a 7 înmulțiri și 18 adunări, pe când metoda clasică necesită 8 înmulțiri și 4 adunări. Într-adevăr, avem următoarea egalitate:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} w & x \\ y & z \end{pmatrix}=\begin{pmatrix} A+B+C-D & D+E \\ F+C & A+E-F+G \end{pmatrix}$$

unde:

$A=(a+d)(w+z)$
$B=(b-d)(y+z)$
$C=d(y-w)$
$D=(a+b)z$
$E=a(x-z)$
$F=(c+d)w$
$G=(c-a)(w+x)$

Acest rezultat poate fi generalizat și în cazul în care a, b, c, d, w, x, y și z sunt, la rândul lor, matrice pătratice și avem astfel posibilitatea de a înmulți pe blocuri matrice al căror ordin este o putere a lui 2. Dacă ordinul este oarecare, se pot completa matricele cu elemente nule până la obținerea unui ordin corespunzător.

Pentru a evalua complexitatea unui algoritm care efectuează produsul a două matrice de dimensiuni $n\times n$ prin această metodă, notăm cu $M(n)$ numărul de înmulțiri și cu $A(n)$ numărul de adunări necesare. Cum metoda presupune efectuarea a 7 înmulțiri și 18 adunări de matrice de ordin $\frac{n}{2}\times\frac{n}{2}$ se observă că putem scrie:

$$M(n)=7M\left(\frac{n}{2}\right)=7^2M\left(\frac{n}{4}\right)=\ldots\Rightarrow M(n)\sim 7^{\log_2 n}=n^{\log_2 7}$$

$$A(n)=18\left(\frac{n}{2}\right)^2+7A\left(\frac{n}{2}\right)\Rightarrow A(n)=\frac{18}{4}n^2\left(1+\frac{7}{4}+\frac{7^2}{4^2}+\ldots\right)\sim\frac{18}{4}n^2\sum_{i=0}^{\log_2 n}\left(\frac{7}{4}\right)^i\Rightarrow$$

$$\Rightarrow A(n)\sim 6n^2\left[\left(\frac{7}{4}\right)^{\log_2 n+1}-1\right]\sim\frac{21}{2}n^{\log_2 7}$$

Deci $t=O(M(n)+A(n))=O(n^{\log_2 7})\approx O(n^{2,81})$, ceea ce constituie o ameliorare în comparație cu metoda clasică, în care $t=O(n^3)$ (mai ales pentru valori mari ale lui n).

Bibliografie:

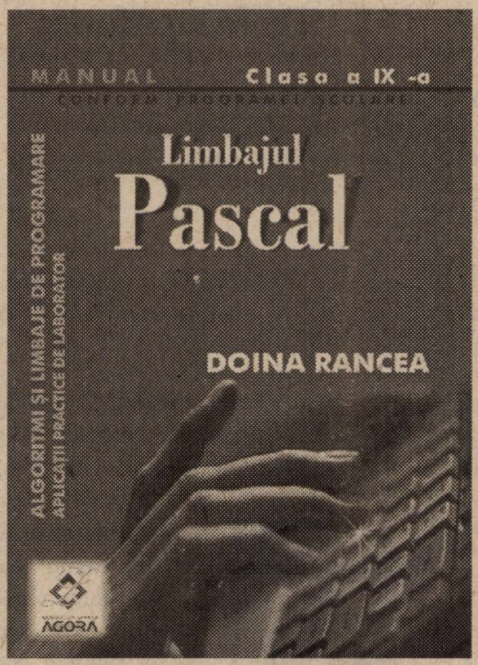
J. C. Boussard, R. Mahl — *Programmation avancée*, Editura Eyrolles, Paris 1987.

D-na Raluca VERNIC este asist. la Universitatea "Ovidius" Constanța.
E-mail: raluca@univ-ovidius.ro

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Limbajul Pascal este un manual care acoperă programa școlară (1997) a două discipline: „Algoritmi și limbaje de programare" și „Aplicații practice de laborator" care se studiază în clasele a IX-a cu profil de informatică.

Manualul prezintă noțiunile din programă la un nivel accesibil, fiecare noțiune fiind ilustrată în mai multe programe Borland Pascal 7.0. Aceste programe sunt însoțite și de prezentarea datelor de intrare și a rezultatelor în forma sub care apar pe ecran. Problemele au fost selectate în așa fel încât să ilustreze multiplele domenii de utilizare a calculatoarelor. Nu lipsesc problemele de simulare, jocuri, probleme distractive, dar au fost alese și aplicații din programa de matematică de clasa a IX-a cu scopul exemplificării și fixării unor noțiuni importante din matematică. În manual se insistă asupra felului în care se descoperă rezolvarea unei probleme, nivelul de dificultate a acestora crescând gradat, venind astfel în sprijinul elevilor și profesorilor.
Numeroasele probleme din manual permit utilizarea acestuia nu numai pentru studiu individual ci și ca bogat material didactic pentru lecțiile de programare și laborator.
Manualul se adresează elevilor, studenților, profesorilor și oricărei persoane interesate de programare.
Manualul a fost conceput, plecând de la ediția din 1993 (Limbajul Turbo Pascal, clasa a X-a) care a fost regândit substanțial și îmbogățit, astfel încât să fie accesibil unui elev de clasa a IX-a. Ediția anterioară s-a bucurat de un succes deosebit, fiind apreciată de mii de cititori.

Autoarea, Doina Rancea, este absolventă cu diplomă de merit a Facultății de Matematică, Secția Informatică a Universității Babeș-Bolyai din Cluj Napoca (1975). A fost repartizată la Liceul de Informatică din Cluj Napoca unde predă și în prezent limbajele de programare Pascal și C++. Este profesoară gradul I și în această calitate îndrumă activitatea de practică pedagogică a studenților. Experiența bogată a autoarei a fost valorificată cu ocazia participării, în cadrul Comisiei Naționale de Specialitate Informatică a Ministerului Educației Naționale, la actualizări ale programei școlare pentru disciplina informatică și în cadrul comisiilor județene și naționale ale olimpiadelor, propunând probleme originale.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Fotbal, SOFTWARE și roboți

De Ionuț Mușlea

După ce un calculator a reușit să învingă la șah pe campionul mondial, urmează o nouă provocare: fotbalul. Deși pare o joacă, proiectul este extrem de serios iar câștigurile pe care le poate aduce în plan tehnologic sunt imense.

Mai sunt cinci secunde de joc și suntem conduși cu 1-0. Suntem cu toții în picioare, cu ochii lipiți de ecran și nu-ndrăznim nici să clipim. Ai noștri intercepteză mingea în preajma propriului careu, pasează precis spre centrul terenului, unde urmează un dribling scurt, o pasă în adâncime spre vârful de atac complet demarcat, care trage "dintr-o bucată" și înscrie incredibil la colțul scurt. Din cele 6 piepturi țâșnește simultan un urlet de bucurie mai ceva decât al unui comentator brazilian: Goooooooooooool!

Deși meciul este în reluare și unii dintre noi îl văd a cincia oară, fețele tuturor s-au luminat și răsuflăm ușurați pentru câteva secunde. Încep prelungirile, portarul nostru intervine salvator, urmează un atac pozițional care nu prevestește nimic spectaculos, dar ai noștri trag surprinzător de la 18-20 de metri, portarul e surprins pe picior greșit și... Goooooool!

Cum jucăm după regula "golului de aur", iată-ne câștigând finala mică a campionatului mondial și plecând acasă cu medaliile de bronz!

RoboCup '97

Deși ați putea crede că rândurile de mai sus reprezintă visul meu legat de Mondialele de la Paris (ce n-aș da să ra "băieții Generalului" bronzul!), sau mai știu eu ce rezultat înregistrat într-o obscură ligă de "pitici", nu este

câtuși de puțin vorba de așa ceva. Meciul chiar a avut loc, s-a jucat pe data de 28 august 1997 la Nagoya (Japonia), și a reprezentat "finala mică" a RoboCup-97, secția "agenți sintetici". Echipa noastră, ISIS (*ISI Synthetic* - University of Southern California), a învins cu 2-1 (1-0, 1-1) pe CMUnited (echipa celor din Carnegie Mellon) și a ieșit pe locul 3 din 32 de echipe.

Robocup-97 a reprezentat una dintre marile atracții ale IJCAI-97 (*International Joint Conference on Artificial Intelligence*), cea mai importantă conferință de inteligență artificială a anului. Robocup-97 a fost prima ediție a campionatului mondial de fotbal pentru roboți/agenți și a avut 3 secțiuni: roboți mici (4 echipe), roboți mijlocii (5 echipe) și agenți sintetici (32 de echipe). Roboții mici au jucat "fiecare cu fiecare", iar cele mai bune două echipe și-au disputat finala, în care Carnegie Mellon a învins echipa japoneză NAIST cu 1-0. Roboții mijlocii au jucat 2 meciuri fiecare, iar cele mai bine clasate două echipe și-au disputat finala. Meciul s-a jucat între DreamTeam (USC) și Osaka University, și s-a terminat la egalitate 0-0. Au urmat prelungirile, care s-au desfășurat până ce roboțeilor li s-au terminat... bateriile. În acel moment a avut loc o discuție extrem de aprinsă, în urma căreia s-a luat decizia de a se decerna titlul de campion mondial *ambelor echipe*.

Practic, competițiile rezervate roboților au fost marcate de limitările

tehnice specifice domeniului. Astfel, roboții se agățau și se blocau reciproc la tot pasul, motorașele se ardeau, bateriile se terminau înainte de vreme, partidele erau întrerupte la 10-15 secunde pentru ca "antrenorii" să-și despartă "odraslele" înțepenite iar unii jucători se încăpățânau să se izbească repetat cu "capul" de primul parapet întâlnit în cale. Mai mult, două dintre cele cinci echipe mijlocii nici nu s-au mișcat din loc pe durata partidelor "disputate", iar punctul forte al meciului între aceste echipe de "statui" (bolovani?) l-a constituit șuieratul prelung al "portarului", care și-a folosit în mod inutil jetul de aer încercând să oprească un șut... inexistent.

În aceeași liga, din cele 6 partide disputate, jumătate s-au încheiat cu un scor alb. DreamTeam-ul a înscris de 8 ori (din care două autogoluri!) în două meciuri (scoruri 4-0 și 2-2). Exceptând DreamTeam-ul, doar echipa din Osaka a mai fost capabilă să înscrie, iar unicul lor gol a fost rodul unui complex de împrejurări cu totul ieșit din comun: la începutul partidei, adversarii lor (australienii de la RMIT) au... uitat să pună un robot în poartă. Lovitura de începere a meciului a fost executată de japonezi, iar întâmplarea a făcut ca primul contact al unui robot cu balonul să-l trimită pe acesta din urma spre poarta nefericiților australieni.

Deși competiția de agenți a fost la un nivel net superior, ea s-a bucurat de o mai mică atenție din partea vizitatorilor. Mulți se așteptau ca roboții să fie echivalenții lui Pele, Maradona, sau măcar a vreunui obscur jucător de "Divizia Onoare", iar frustrarea inițială a creat o stare de spirit în care lumea s-a mulțumit cu luatul la mișto a "vajnicilor combatanți". Revenind acum la agenți, trebuie să spunem că această competiție a fost mult mai apropiată de echivalentul ei uman, atât ca sistem de disputare, cât și din punct de vedere al calității jocului. Cele 32 de echipe au fost împărțite în 8 grupe, în care fiecare echipă le-a întâlnit într-un meci direct pe celelalte trei. Primele două clasate s-au calificat pentru faza următoare, care s-a jucat în sistem

Scurtă istorie

Ideea ca roboții să joace fotbal nu aparține – așa cum s-ar putea crede – lui Isaac Asimov, ci profesorului Alan Mackworth de la Universitatea British Columbia (Canada). A fost prima dată exprimată într-un articol apărut în 1992 și reluată un an mai târziu în cartea sa *"Computer Vision: System, Theory, and Applications"*.

Cam în aceeași perioadă, în Japonia, un grup de cercetători au început să studieze beneficiile pe care simularea jocului de fotbal le-ar putea aduce în promovarea științelor și tehnologiilor asociate cu inteligența artificială și robotică.

În scurt timp se trece la fapte: un grup de cercetători în frunte cu Minoru Asada inițiază proiectul unei ligi japoneze de fotbal a roboților iar un centru de cercetări guvernamental japonez (ETL – *ElectroTechnical Laboratory*) lansează un proiect de cercetare în domeniul sistemelor multi-agent și începe dezvoltarea unui simulator pentru jocul de fotbal.

Ca urmare a interesului stârnit de proiect în mediile științifice din afara Japoniei, proiectul devine internațional și capătă numele *Robot World Cup Initiative*, pe scurt RoboCup.

În septembrie 1993, când apare primul anunț al inițiativei, câteva proiecte în acest domeniu erau deja lansate, printre care cel condus de profesorul Minoru la universitatea din Osaka precum și un proiect similar la Carnegie Mellon University.

În scurt timp, ETL finalizează SoccerServer versiunea 0 (scris în Lisp), un sistem deschis care permite simularea jocului de fotbal cu sisteme multi-agent.

Proiectul este discutat la *National Conference in Artificial Intelligence* (AAAI-94) la Seattle.

În 1995 este realizat SoccerServer 1.0 (de data aceasta în C++) și are loc prima demonstrație publică cu ocazia *International Joint Conference on Artificial Intelligence* (IJCAI-95), la Montreal. Tot atunci se decide organizarea primei ediții RoboCup la Nagoya, în conjuncție cu IJCAI-97, precum și a unei ediții Pre-RoboCup-96, în ideea de a identifica în această ediție de test problemele care ar putea să apară.

Pre-RoboCup-96 are loc la Osaka, cu ocazia International *Conference on Intelligence Robotics and Systems* (IROS-96). Participă opt echipe la clasa simulare și au loc demonstrații cu roboti de clasa mijlocie.

La prima ediție oficială, RoboCup-97, participă 40 de echipe și peste 5.000 de spectatori.

Anul acesta, la Paris, sunt așteptate cca. 100 de echipe. Se estimează că va fi cea mai mare întâlnire de roboți mobili din istorie.

(M.S.)

eliminatoriu: optimi, sferturi, semifinale și cele două finale.

Exceptând jocurile din grupe, când "agenții-statui" au fost snopiți de ceilalți cu scoruri de nivelul 25-0, restul meciurilor au fost extrem de echilibrate, în ciuda unor diferențe de scor aparent enorme (11-2 în finală, 14-0 și 6-0 în semifinale). Fiecare echipă a constat din 11 jucători (adică 11... programe), care primeau informațiile despre terenul de joc de la un SoccerServer ce a jucat rolul de coordonator central (serverul a fost implementat de către organizatori iar interfața agent-server a făcut obiectul unei dezbateri publice). Meciurile au fost vizualizate într-o fereastră de culoare verde, "liniată" ca un teren de joc pe care agenții erau reprezentați sub formă de cerculețe. Vizionarea unui astfel de meci e asemănătoare cu cea a unei partide de "fotbal cu nasturi" privită de sus, cu unica diferență că jucătorii nu se mișca unul câte unul, ci toți deodată.

Regula jocului

Interfața server-agent este deosebit de simplă, comunicarea desfășurându-se după cum urmează: la intervale 300 milisecunde, fiecare agent/program transmite serverului acțiunea pe care dorește să o execute (de exemplu: *sprintează*, *trage*, *rotește-te*, etc.), iar serverul execută acțiunile în cauză și actualizează situația pe terenul de joc. Reciproc, la intervale de 100 milisecunde, serverul transmite fiecărui agent informația vizuală (în funcție de orientarea agentului și de poziția sa pe teren, la un moment dat diversele obiecte pot fi vizibile, parțial vizibile sau invizibile) și auditivă (agenții pot transmite mesaje care "se aud" până la o anumită distanță). Pentru crearea unor situații cât mai realiste, serverul folosește și parametri de tipul "viteza vântului" sau "factori aleatori", cum ar fi efectul neașteptat al unui smoc de iarba ori a unei bălți aflate în calea mingii.

Poate că cea mai bună și concisă descriere a scopului RoboCup este data de Kitano et al. în "abstract"-ul articolului "The RoboCup Synthetic Agent Challenge 97":

"RoboCup Challenge oferă un set de probleme deschise legate de sisteme multi-agent care acționează în timp real în medii dinamice. Deși RoboCup-ul a fost gândit ca o problemă-test pe termen lung (două sau chiar trei decenii), RoboCup Challenge propune trei probleme concrete pentru următorii doi ani: învățarea de către agenți individuali și în echipă, planning și executarea planurilor în sisteme multi-agent, și modelarea oponentului".

Hackeri, blaturi și aranjamente

Acum un an, RoboCup Challenge părea un vis frumos, căci competiția pre-Robocup (8 echipe participante)

a fost câștigată detașat de echipa unui student de la Tokyo University, care a hack-uit de unul singur o echipă ce n-avea nimic în comun cu inteligența artificială și se baza exclusiv pe un noian de calcule trigonometrice. Poate că cel mai mare succes al RoboCup-97 a constat în faptul că echipele hacker-ilor au fost complet "măturate" de cele ale "savanților": nu numai că primele patru clasate au la baza un serios fundament științific, dar, mai mult, nici unul dintre hackeri nu a trecut de faza optimilor! Și asta în condițiile în care înaintea competiției se zvonea că hackerii niponi sunt atât de puternici încât vor trebui "sa aranjeze" (din milă!) ca și o echipa ne-japoneză "să prindă" semifinalele.

Evident, nici competiția agenților n-a fost ferită de surprize neplăcute: unele universități au trimis două echipe șupărător de asemănătoare, iar alte echipe au folosit trucuri nu tocmai ortodoxe. Spre exemplu, unii au încercat să "back-engineer" Soccer Server-ul, alții au obstrucționat jocul încercând să încarce traficul de rețea prin transmiterea unei avalanșe de mesaje auditive, iar o altă categorie de "șmecheri" înregistra și retransmitea mesajele transmise cu puțin înainte de către oponenți. În fine, deși valoarea campioanei mondiale (Humboldt University, nemții ăștia-s dați naibii!) este absolut evidentă, victoriile lor s-au bazat în bună măsură pe un bug pe care l-au găsit în server și, spre deosebire de bug-urile descoperite de alți participanți, nu l-au făcut public. În două cuvinte, e vorba de faptul că serverul permite (în mod eronat!) lovirea mingii de mai multe ori "în același moment", iar acest lucru duce la obținerea unor șuturi mult mai puternice decât cele permise în mod normal. Pe baza acestui truc, teutonii înscriu imparabil chiar și din propria jumătate de teren, căci mingea atinge viteze care o fac de ne-interceptat!

Urmează Parisul

Dar viața merge înainte, RoboCup-97 e deja de domeniul trecutului iar lumea se pregătește febril Robocup-98, care va avea loc la Paris în timpul Mondialelor "adevărate". Ar fi nemaipomenit să fie prezente acolo și echipe românești (informații se pot obține direct din pagina lui Hiroaki Kitano,http://www.csl.sony.co.jp/person/kitano/RoboCup/RoboCup.html), și mi-e greu să-mi doresc altceva decât o finală în care ISIS-ul să întâlnească pe, să zicem, ...Poli Cluj.

Ionuț Mușlea este doctorand la University Of Southern California. E-mail: muslea@isi.edu.

COMPUTER PRESS
AGORA

Prin intermediul firmei noastre puteți primi în mod regulat oricare dintre revistele americane de calculatoare din tabelul alăturat. Pentru aceasta este suficient să ne trimiteți o comandă în care să specificați titlurile dorite și datele Dvs. de identificare (nume, adresă, telefon, fax), iar noi vă vom expedia cu plata prin ramburs revistele solicitate. Prețul în lei pe care îl veți achita la primirea fiecărui colet poate fi obținut prin înmulțirea prețului în dolari specificat în tabel cu cursul de schimb valutar al BNR la care se adaugă cheltuiala de expediție percepută de Poșta Română. Atenție deci,

NU TRIMITEȚI BANI ÎN AVANS!!!

Vă rugăm să expediați comenzile Dvs. pe adresa:

Computer Press Agora srl, CP 94 OP 49, București, fax: 3309285

TITLU	FRECVENȚĂ	PREȚ
3D DESIGN	12	5.93
ACCESS OFICE VBA ADVISOR	12	7.49
AIXTRA	4	7.43
BOARDWATCH	12	8.93
BYTE	12	5.93
C/C++ USERS JOURNAL	12	7.43
C++ REPORT	9	11.93
UNICENTER TNC ADVISOR	6	8.93
CADALYST	12	5.93
CADENCE	12	5.93
CD-ROM PROFESSIONAL	12	10.43
CIRCUIT CELLAR	12	5.93
COMPUTER ARTIST	6	7.43
COMPUTER GAMING WORLD	12	5.99
COMPUTER GAMING WORLD wCD	12	11.99
COMPUTER GRAPHICS WORLD	12	7.43
COMPUTER PLAYER	12	10.43
COMPUTER SHOPPER	12	7.49
COMPUTER SHOPPER BUYERS CD		7.49
CYBER SPORTS	4	7.49
DATA BASE WEB ADVISOR	12	7.49
DATABASE PROG. & DESIGN	12	7.43
DBMS	12	5.93
DBMS 1996 BUYERS GUIDE		7.43
DELPHI INFORMANT	12	8.93
DIGITAL VIDEO MAGAZINE	12	7.43
DR. DOBBS JOURNAL	12	7.43
ELECTRONIC PUBLISHING	12	7.43
ELECTRONIC GAMING MONTH	12	7.49
EMG2	12	7.49
FILE MAKER ADVISOR	24	7.99
FOXPRO ADVISOR	12	7.49
GAMEFAN	12	8.93
GAME DEVELOPER	6	8.93
ULTRA GAME PLAYER	12	7.49
GAME PRO	12	7.49
INTERACTIVITY	12	7.43
INTERNET WORLD	12	7.43
INTERNET JAVA & ACTIVEX ADVISOR	6	10.49

TITLU	FRECVENȚĂ	PREȚ
JAVA DEVELOPERS JOURNAL	12	11.93
JAVA PRO	4	10.43
JAVA REPORT	6	7.43
LAN MAGAZINE	12	7.43
LINUX JOURNAL	12	6.00
LOTUS NOTES ADVISOR	12	10.49
MACADDICT	12	11.99
MACTECH	12	8.98
MACWORLD	12	8.98
MAGAFAN	6	5.93
MICROSOFT INTERACTIVE DEV	12	7.43
MICROSOFT SYSTEMS JOUR.	12	7.43
MONDO 2000	4	8.93
NEW MEDIA	12	7.43
OBJECT MAGAZINE	6	6.75
OBJECT ORIENTED PROGRAM.	9	13.50
PC COMPUTING	12	5.99
PC GAMER	12	5.99
PC GRAPHICS & VIDEO	12	7.43
PC MAGAZINE	22	5.99
PC MAGAZINE CD-ROM	4	26.93
PC MAGAZINE SPECIAL ISSUE	4	8.93
PC WORLD	12	8.93
PRACTICAL WINDOWS	6	8.93
PUBLISH	12	7.43
SOFTWARE DEVELOPMENT	12	5.93
SYS ADMIN	6	7.43
UNIX REVIEW	12	5.93
VB TECH JOURNAL	12	5.25
VIDEO TOASTER USER	6	7.43
VISUAL BASIC PROG. JOUR.	12	5.93
VISUAL DEVELOPER MAGAZINE	6	7.43
WEB, THE	12	5.93
WEB DEVELOPER	6	8.93
WEB TECHNIQUES	12	7.43
WINDOWS/DEV DEVELOPERS	12	7.43
WINDOWS NT	12	7.43
WINDOWS SOURCES	12	4.49
WINDOWS TECH JOURNAL	12	6.00
WIRED	12	7.43

Revistele pot fi răsfoite, comandate și cumpărate de la librăria noastră deschisă la sediul firmei ***Forte Computers****: str. Lipscani nr. 102; precum și de la următoarele magazine din* ***București****: Flamingo Computers, Bd.N. Titulescu nr.102; MIVAS: Pasaj Universitate; Raffles Computers Shop, Calea Victoriei nr.25; Librăria "Noi", sala Dales; Librăria Humanitas din Pasajul Cretzulescu (lîngă magazinul Muzica);* ***BRAȘOV****: Interprof International, str.Operetei nr.86;* ***TIMIȘOARA****: XCOMP21 SRL Bul Regele Ferdinand nr. 2*

Puteți comanda orice carte de la: ***Computer Press Agora s.r.l.****, C.P. 94 OP 49, București, fax: 01-3309285*

Important: Lista completă de prețuri poate fi obținută via BBS 065-210780 (login: bbs, terminal: ANSI, ...)

JAVA – a doua LECȚIE

de Horia Georgescu

Continuăm în acest număr prezentarea limbajului Java. Reamintim că deocamdată ne vom îndrepta atenția spre elementele care ne vor convinge că putem realiza în Java cel puțin ceea ce putem realiza în Pascal sau C. De aceea atenționăm cititorii că serialul de față se adresează celor care au deja cunoștințe minime de programare și sunt familiarizați cu tehnicile de programare fundamentale.

Din nou despre liste

Programul următor implementează metoda sortării cu ajutorul arborilor de sortare. Este folosită metoda `input` descrisă în numărul anterior.

Esența programului este cuprinsă în clasa `elem`.

Câmpurile sunt: variabila întreagă `c` și referințele `st` și `dr` de tip `elem`, corespunzătoare unei înregistrări asociate unui vârf, care memorează valoarea asociată vârfului și adresele descendenților stâng și drept.

Constructorul `elem` inițializează câmpurile obiectului care a produs invocarea sa cu valoarea asociată și cu referințele nule, adică este creată o frunză în arbore.

Metoda `adaug` primește referința către un vârf și o valoare întreagă. Este creată o nouă frunză `x` cu ajutorul constructorului `elem` și este întoarsă "adresa" ei astfel încât frunza să devină descendentul stâng sau drept al unei frunze din arborele de rădăcină `x`. Determinarea vârfului la care este "atârnată" noua frunză se face recursiv printr-o parcurgere a arborelui de rădăcină `x`, conform definiției arborilor de sortare (pentru orice vârf subarborele stâng conține valori mai mici sau egale cu ale vârfului, iar subarborele drept conține valori mai mari sau egale cu ale vârfului considerat).

Metoda `parcurg` întoarce un șir ce conține, în inordine, valorile din subarborele de rădăcină `x` (parametrul metodei).

Clasa principală `arbsort` efectuează, prin metoda `main`, următoarele prelucrări:

- este citit un număr întreg și este creat arborele (frunza) ce conține acea valoare; "adresa" sa este întoarsă în `rad`;
- atâta timp cât la intrare urmează valori numerice (prima valoare nenumerică este detectată cu ajutorul funcției `isNaN` = is Not a Number), ele sunt citite și adăugate arborelui de sortare curent;
- după ce au fost citite toate numerele ce trebuie ordonate crescător, se efectuează o parcurgere în inordine a arborelui de rădăcină `rad`, prin invocarea metodei `parcurg` pentru obiectul `rad`.

Pentru o mai bună înțelegere, să presupunem că obiectul `rad` are ca descendent stâng obiectul `rads`, iar ca descendent drept obiectul `radd`. În cazul adăugării unei valori `val` în arborele de rădăcină `rad`, lucrurile se petrec după cum urmează. Obiectul `rad` este pus să execute metoda sa `adaug`. Dacă `val` este mai mic sau egal cu câmpul numeric al obiectului `rad`, atunci `rad` "comandă" obiectului `rads` să execute metoda sa `adaug`, adică să insereze valoarea `val` în subarborele de rădăcină `rads`. Cazul complementar este tratat analog.

```
import java.io.*; import input;
class elem {
  int c; elem st,dr;

  elem(int ch)  { c=ch; st=null; dr=null; }

  elem adaug(elem x, int ch) {
    if (x==null) x=new elem(ch);
    else if (ch<x.c) x.st=adaug(x.st,ch);
         else x.dr=adaug(x.dr,ch);
    return x;
  }

  String parcurg(elem x) {
    if (x==null) return "";
    else return( parcurg(x.st) + x.c
                 + " " + parcurg(x.dr));
  }
}
```

```
class arbsort {
  public static void main(String arg[]) {
    elem rad=null; double val;
    StreamTokenizer f =
        new StreamTokenizer(System.in);
    val = input.read(f); rad = new elem( (int) val);
    val = input.read(f);
    while ( ! Double.isNaN(val) ) //  ! este negatia
      { rad.adaug(rad,(int) val);
        val = input.read(f); };
    System.out.println(rad.parcurg(rad));
    input.readln();
  }
}
```

Observăm că sunt respectate principiile programării orientate spre obiecte: clasa `elem` "pune la un loc" câmpuri și operații (metode) prin care aceste câmpuri sunt prelucrate.

Programul de mai sus se pretează la o mică (și incompletă) discuție asupra noțiunii de *tip* de date.

Să considerăm de exemplu tipul de bază **int**. În multe locuri, chiar acest tip simplu este prezentat incorect: se reduce noțiunea de tip la mulțimea valorilor posibile. În realitate este vorba nu numai de această mulțime, ci și de prelucrările ce se pot efectua asupra valorilor de tipul respectiv (adunare, scădere etc.).

Lucrurile devin mai complexe pentru clase. O clasă este un tip caracterizat prin:

- câmpuri (cu tipul lor);
- metode (cu tipul lor);
- drepturile de acces la fiecare câmp și metodă;
- drepturile de moștenire referitoare la câmpuri și metode.

Tablouri

În paragraful anterior am prezentat un exemplu de utilizare a vectorilor (tablouri unidimensionale). Prezentăm în continuare noi elemente legate de tablourile uni- și multidimensionale.

Tablourile sunt colecții de date de același tip.

Un tablou unidimensional `a` poate fi declarat prin:

```
tip[] a;
```

unde `tip` este tipul componentelor tabloului.

Crearea unui astfel de tablou se poate face prin:

```
tip a = new tip[n];
```

unde `n` este o constantă sau o variabilă întreagă strict pozitivă. Elementele tabloului pot fi referite prin `a[i]` cu `i` în intervalul `0..n-1`, iar lungimea tabloului poate fi referită în continuare prin `a.length`.

Declararea și crearea pot fi făcute simultan prin:

```
tip[] a = new tip[n];
```

sau printr-o inițializare efectivă, ca de exemplu:

```
int[] a = {0,3,2,5,1}
```

prin care, evident, `a.length` devine 5.

Pe parcursul unui program, un tablou poate fi redefinit:

Elemente

Caractere

Limbajele de programare "clasice" utilizează setul de caractere ASCII. Limbajul Java folosește setul de caractere *Unicode*, care este reprezentabil pe 16 biți, deci are 65.536 caractere, cuprinzând caractere și simboluri proprii aproape tuturor limbilor existente.

Primele 256 caractere din Unicode sunt cele din setul de caractere *Latin-1*, dintre care primele 128 sunt cele din *ASCII*.

Un caracter poate fi reprezentat oriunde în textul sursă printr-o secvență Escape, având forma \uhhhh sau \Uhhhh, unde am notat prin h o cifră hexazecimală (cifrele hexazecimale corespunzătoare lui 10-15 sunt identificate prin a-f sau A-F). Deci, o secvență Escape poate apare nu numai în caractere și șiruri de caractere, dar și în identificatori. Numărul hexazecimal dintr-o astfel de secvență corespunde numărului de ordine al caracterului în setul Unicode.

Ne mărginim la a indica secvențele Escape pentru semnele diacritice din limba română:

ă = \u00c3	ă = \u00e3
â = \u00c2	â = \u00e2
ș = \u00aa	ș = \u00ba
ț = \u00de	ț = \u00fe
î = \u00ce	î = \u00ee

În continuare vom înțelege prin *spații albe* următoarele caractere: blanc, tab, newline, form-feed. Anumite caractere pot fi reprezentabile printr-o secvență Escape de o formă aparte:

\n = \u000a (linie nouă) . . .\\ = \u005c (backslash)
\t = \u0009 (tab) . . . . . .\' = \u0027 (apostrof)
\b = \u0008 (backspace) .\f = \u000c (form-feed)
\r = \u000d (return) . . . . .\" = \u0022 (ghilimele)

Motivația utilizării setului Unicode este integrarea în limbaje de tip HTML care să permită circulația pe Internet.

Identificatori

În Java identificatorii încep cu o literă (inclusiv '_' și '$') și continuă cu litere sau / și cifre. Literele și cifrele extind pe cele obișnuite incluzând și pe cele din alte limbi. Un identificator poate avea orice lungime; de aceea este indicat să folosim identificatori cât mai sugestivi pentru rolul pe care îl joacă, timpul "pierdut" la scrierea programului fiind recuperat din plin atunci când revedem programul peste câteva luni sau când trebuie ca el să fie înțeles de alte persoane.

Anumiți identificatori au o destinație precisă ce nu poate fi modificată, deci nu pot fi folosiți în alt context decât cel intenționat de către autorii limbajului. Ei se numesc *cuvinte cheie*. O primă listă de

```
tip[] a ;
tip[] a = new tip[10];
......
tip[] a = new tip[20];
......
```

Este posibil să definim și să creăm tablouri de obiecte, bineînțeles de același tip (instanțe ale aceleiași clase), tablouri de șiruri de caractere etc.

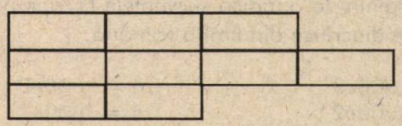
Tablourile multidimensionale trebuie gândite ca tablouri unidimensionale ale căror elemente sunt tablouri unidimensionale etc. De aceea referirea la un element al unui tablou multidimensional a se face prin:

`a[indice]...[indice].`

Este suficient să reducem discuția la tablouri bidimensionale, generalizarea fiind imediată. Să considerăm următorul exemplu:

```
int[][] a = new int[3][];
a[0] = new int[3];
a[1] = new int[4];
a[2] = new int[2];
```

Ia naștere astfel un tablou de forma:



ceea ce arată că în Java tablourile nu sunt neapărat dreptunghiulare; aceasta conduce desigur la economie de spațiu.

Evident, `a[1].length=4`.

La aceeași structură se poate ajunge și printr-o inițializare efectivă:

```
int[][] a = { {0,1,2}, {1,2,3,4}, {2,3} };
```

care în plus atribuie și valori elementelor tabloului.

Dacă în referirea la un element al unui tablou unul dintre indici nu este în intervalul corespunzător, va apărea eroarea (numită în Java *excepție*) cu numele `IndexOutOfBoundsException`.

Reamintim că tablourile pot fi redefinite. Astfel, pentru exemplul de mai sus poate apare în continuare:

```
a[1] = new int[7];
```

ceea ce permite să schimbăm "o parte" dintr-un tablou fără a defini un altul și a copia părțile comune.

Programul care urmează implementează algoritmul DF de parcurgere în adâncime a grafurilor neorientate (vezi G.I. nr. 2 , 1996).

Clasa principală este `ParcDF`, deci fișierul sursă va avea numele `ParcDF.java`. Metoda `main` crează un obiect `Ob` al clasei `df`, iar apoi îl folosește pentru invocarea metodelor `listare` și `parc` ale acestei clase.

Este folosită clasa `input` prezentată în numărul precedent.

Clasa `df` utilizează în principal următoarele câmpuri:

- `n` = numărul de vârfuri din graf;
- `mat` = matricea care va conține lista vecinilor fiecărui vârf;
- `nrdf` = vector care va memora pentru fiecare vârf `i` numărul său de ordine în parcurgerea DF;
- `ndf` = ultimul număr atașat unui vârf în parcurgerea DF.

Aceste câmpuri sunt declarate cu modificatorul **`static`** pentru că sunt comune tuturor obiectelor clasei `df`.

Constructorul clasei nu are nici un parametru și are rolul de a face inițializări. Se începe prin citirea numărului `n` de vârfuri ale grafului. Este creat vectorul `nrdf` de dimensiune `n` și vectorul `mat` de aceeași dimensiune, ale cărui elemente sunt vectori. În continuare, pentru fiecare vârf `i` se citesc vecinii săi până când este detectat la intrare altceva decât un număr. Pentru aceasta la intrare lista vecinilor unui vârf se va termina (de exemplu) cu o literă; prin program acest "delimitator" este detectat de funcția standard `Double.isNaN`. Numărul vecinilor lui `i` este calculat în `nrelem`, iar vecinii sunt citiți în primele `nrelem-1` componente ale lui `nrdf`, care este folosit aici ca vector de lucru. Apoi este creat vectorul `mat[i]` de lungime `nrelem` și sunt mutate în el primele `nrelem-1` componente ale lui `nrdf`.

Este evident că tabloul bidimensional `mat` nu este în general un pătrat, numărul elementelor de pe fiecare linie putând diferi de la o linie la alta.

Metoda `parc` mai folosește variabila `cc` a cărei valoare reprezintă numărul ultimei componente conexe determinate. Este folosită și apelată recursiv metoda `DF`, a cărei acțiune este cea din referința citată.

Să observăm că din afara clasei `df` știm doar că ea conține metodele `listare` și `parc` fără nici un argument. În acest mod toată esența problemei este încapsulată și ascunsă în clasa `df`, urmându-se unul dintre imperativele programării orientate spre obiecte.

```
import java.io.*; import input;

class df {
   static StreamTokenizer f =
          new StreamTokenizer(System.in);
   static int[][] mat; static int[] nrdf;
   static int n,ndf,cc; static double d;
   int nrelem,i,j;

   df() {
      n = (int) input.read(f);
      mat = new int[n][]; nrdf = new int[n];
      for (i=0; i<n; i++) {
         nrelem = 0; d = input.read(f);
            while(! Double.isNaN(d))
            { nrdf[nrelem++]= (int) d;
              d = input.read(f); }
         mat [i] = new int[nrelem];
         for (j=0; j<nrelem; j++)
            mat[i][j] = nrdf[j];
      }
      System.out.println("**************");
   }
   void listare() {
```

```
        for (i=0; i<n; i++) {
            System.out.print(i + " :\t");
            for (j=0; j<mat[i].length; j++)
                System.out.print(mat[i][j]+" ");
                System.out.println();
        }
        System.out.println("**************");
    }

    void parc() {
        ndf=0; cc=0; for (i=0; i<n; i++) nrdf[i]=0;
        for (i=0; i<n; i++)
            if (nrdf[i]==0) {
                System.out.print("Componenta conexa "
                            + ++cc + " :\t");
                DF(i); System.out.println();
            }
        input.readln();
    }

    void DF(int i) {
        int j,k;
        nrdf[i] = ++ndf; System.out.print(i + " ");
        for (j=0; j<mat[i].length; j++)
            { k=mat[i][j]; if (nrdf[k]==0) DF(k); }
    }
}

class ParcDF {
    public static void main(String[] s) {
        df Ob = new df(); Ob.listare(); Ob.parc();
    }
}
```

Referința *this*, câmpuri statice și blocuri de inițializare

Într-o metodă (a unei clase), care nu este declarată cu modificatorul **static**, putem utiliza referința obiect **this** pentru a ne referi la obiectul curent, adică cel care a invocat metoda. Prezentăm doar două situații în care această referință specială este de obicei folosită:

1) *în interiorul constructorilor*:

```
class clasa {
  ... int i; ...
  clasa(int i) { this.i=i;}
  ...
}
```

prin care câmpul `i` al clasei `clasa` primește valoarea `i` transmisă la crearea obiectului prin utilizarea constructorului de mai sus. În acest mod este clară distincția între câmpul `i` și parametrul `i`. Evident, am putea renumi parametrul:

```
clasa (int ii) { i=ii; }
```

ceea ce poate conduce însă uneori la confuzii privind semnificația variabilelor.

cuvinte cheie este următoarea:

boolean	**break**	**byte**	**catch**	**char**	**class**
continue	**do**	**double**	**else**	**final**	**float**
for	**if**	**import**	**int**	**long**	**new**
null	**private**	**public**	**return**	**short**	**static**
synchronized		**switch**	**try**	**void**	**while**

În prezentarea pe care o facem, în programe aceste cuvinte cheie vor apare îngroșate pentru a veni în ajutorul cititorilor.

Constante

În limbajul Java, constantele mai poartă și numele de *literali*. Vom prezenta doar o parte a acestor constante:

1) *Literalii întregi* sunt de două tipuri:

• *normali*, reprezentați pe 32 de biți și pentru care scrierea este cea uzuală;

• *lungi*, reprezentați pe 64 de biți, identificați prin faptul că au sufixul 'l' sau 'L'.

2) *Literalii reali*:

• *dubli*, repezentați pe 64 de biți, pentru care scrierea este cea uzuală. Este vorba de numere zecimale ce conțin opțional punctul zecimal și pot fi urmate de un exponent prefixat cu e sau E. De exemplu următorii trei literali reprezintă aceeași valoare: 21. 2.1e1 .21E2

• *normali*, reprezentați pe 32 de biți și pentru care sufixul este 'f' sau 'F'.

3) *Literalii booleeni* sunt true și false.

4) *Literalii de tip caracter*. Ne vom restrânge la cei cu reprezentare grafică, având forma 'c' unde c este un caracter sau o seqvență Escape predefinită; aceste secvențe au fost prezentate mai sus.

5) *Literalii de tip șir de caractere* sunt cuprinși între ghilimele.

Menționăm că șirurile de caractere sunt instanțe ale clasei **String** declarată standard în pachetul java.lang. Este permis operatorul de concatenare +. Iată câteva exemple: "" (șirul vid); "ab\nn" (unde apare secvența Escape '\n'); "unu"+"doi"

Comentarii

Menționăm următoarele modalități de specificare:

• de sfârșit de linie: încep cu // și se termină la sfârșitul liniei;

• generali: încep cu /* și se termină cu */.

Nu există comentarii în comentarii.

Variabile

Forma generală folosită pentru declararea variabilelor este:

<șir de modificatori> <tip> <listdeidentificatori>;

unde prezența modificatorilor este opțională. Informațiile din șirul de modificatori sunt separate prin spații albe, iar cele din lista de identificatori prin virgulă. Identificatorii pot fi însoțiți de specificarea unei valori inițiale prin sufixarea lor cu semnul = urmat de o valoare, ca de exemplu în: **int** a = 3;

Modificatorii se împart în 4 categorii:

• modificatori de acces (**public**,**private**,**protected**);

• **static** ;

• **synchronized** ;

• **final** .

2) *la apelarea unei metode, când dorim să transmitem ca parametru obiectul curent.* Aceasta este util de exemplu când vrem să creăm un nou obiect care să păstreze referința către cel curent:

```
class Clasa {
  int i,j; Clasa Ob;
  ...
 Clasa(int i, Clasa Ob) {this.i=i; this.Ob=Ob;}
  ...
     Ob1=new Clasa(j,this);
  ...
}
```

Un al doilea aspect la care ne vom referi în acest paragraf este posibilitatea de a declara membri (câmpuri și metode) ale unei clase cu modificatorul **static**; vom spune că acești membri sunt *statici* sau *de clasă*. Ne vom mărgini deocamdată la a vorbi despre câmpurile statice.

Într-o primă aproximație, putem considera că la crearea unui nou obiect ca tip al unei clase, vor fi create:

- dubluri ale câmpurilor clasei;
- referințe către codul corespunzător metodelor.

Apar însă nu puține situații în care mai multe obiecte ale aceleiași clase folosesc mai multe câmpuri drept "memorie comună", fie pentru că aceste câmpuri nu se modifică, fie pentru că ele sunt actualizate atunci când obiectele execută metode ale clasei și obiectele folosesc valoarea acestor câmpuri pentru a decide asupra modului în care vor continua calculele. În asemenea situații soluția este de a declara câmpurile respective ca fiind statice.

Un câmp static (de clasă) poate fi referit din afara clasei prin simpla sa prefixare cu numele clasei.

Un al treilea aspect de care ne vom ocupa în acest paragraf îl constituie *blocurile de inițializare*.

O clasă poate avea unul sau mai multe blocuri de inițializare. Ele trebuie declarate cu modificatorul **static**:

```
static { ... }
```

În principal, blocurile de inițializare sunt folosite pentru inițializarea câmpurilor statice ale clasei. În exemplul următor:

```
class C {
  static int[] a = new int[10];
  static {
    for (int i=0; i<a.length; i++) a[i]=0;
  }
  ...
}
```

componentele vectorului a au fost inițializate cu zero.

Inițializările statice, de câmpuri și/sau de blocuri de inițializare, se fac în ordinea în care apar acestea în cadrul clasei. Astfel, pentru exemplul de mai sus este clar că blocul de inițializare trebuie *precedat* de crearea vectorului a.

Vom ilustra noțiunile introduse mai sus pentru a produce un program ce rezolvă următoarea problemă:

Se dă un caroiaj dreptunghiular cu `m` *linii și* `n` *coloane. Unele dintre celulele caroiajului sunt libere, altele sunt ocupate. Un mobil se află în poziția inițială* `(i0,j0)`*; el se poate deplasa numai către pozițiile din stânga, dreapta, sus și jos, evident dacă acestea sunt libere și nu se iese din caroiaj. Se cere să se determine toate drumurile prin care mobilul poate ajunge la frontieră.*

Să presupunem că respectivul caroiaj are configurația:

```
1 1 1 1 0 1
1 0 0 0 0 1
1 0 0 0 0 1
1 1 1 1 0 1
```

Vom borda caroiajul cu celule având valoarea 2:

```
2 2 2 2 2 2 2 2
2 1 1 1 1 0 1 2
2 1 0 0 0 0 1 2
2 1 0 0 0 0 1 2
2 1 1 1 1 0 1 2
2 2 2 2 2 2 2 2
```

și vom considra că `m=6`, `n=8`. Valorile lui `i0` și `j0` rămân neschimbate, ținând cont că pentru memorarea câmpurilor caroiajului vom folosi o matrice (pentru care numerotarea liniilor și coloanelor începe cu 0).

Pentru rezolvarea problemei vom utiliza metoda backtracking. Starea mobilului la fiecare moment de timp va fi caracterizată de:

- poziția `(i,j)` a mobilului;
- starea anterioară în care s-a aflat mobilul.

Drept urmare vom construi o clasă `elem` în care sunt esențiale câmpurile:

- `i` și `j` de tip **int**, memorând o poziție în cadrul configurației;
- `prec` de tipul `elem`, memorând starea anterioară a mobilului.

Clasa va mai conține următoarele câmpuri statice:

- `m,n,i0,j0` cu semnificația de mai sus;
- matricea `depl` cu două linii, care memorează incrementii corespunzători deplasărilor. În cazul nostru, matricea `depl` va avea forma:

```
1 0 -1 0
0 -1 0 1
```

corespunzătoare deplasărilor spre S, V, N, E; numărul de deplasări (numărul de coloane al matricii `depl`) va fi memorat în variabila statică `ndepl`, care primește valoarea corespunzătoare prin acțiunea unui bloc de inițializare;

- matricea `mat`, care memorează configurația curentă a caroiajului.

Un prim constructor al clasei nu are nici un parametru și realizează citirea configurației inițiale a caroiajului și a valorilor `i0` și `j0`; este citită partea "efectivă " a caroiajului și se adaugă o primă și ultimă linie și coloană, ale căror elemente primesc toate valoarea 2.

Un al doilea constructor al clasei are 3 parametri și

anume o poziție (i,j) și un obiect prec de tipul elem.

Metoda print are ca parametru un obiect x de tipul elem. Ea întoarce un șir de caractere ce constituie drumul prin care s-a ajuns de la (i0,j0) la (i,j). Sunt necesare câteva explicații. Metoda va fi apelată în momentul în care s-a ajuns cu parcurgerea la frontieră (într-o celulă având valoarea 2). La acel moment obiectul curent va avea valori pentru câmpurile i,j,prec. Listarea drumului se va face prin invocarea System.out.println(print(prec)). Corespunzător, metoda print va întoarce x.print (x.prec), adică drumul până la celula anterioară, la care se adaugă poziția curentă (i,j). Ieșirea din recursivitate se va face în momentul în care s-a ajuns la (i0,j0), situație în care prec va avea valoarea **null** (vezi programul principal).

Metoda back are parametrul prec de tip elem, semnificând starea anterioară. Se încearcă deplasarea spre o nouă poziție (ii,jj). Sunt posibile următoarele situații:

- mat[i][j]=1 : nu se poate avansa, noua poziție fiind ocupată;
- mat[i][j]=2 : se trece la regăsirea drumului prin invocarea metodei print;
- mat[i][j]=0 : se poate avansa. în acest caz mat[i][j] primește valoarea -1 (pentru a nu se intra într-un ciclu infinit), este creat un nou obiect memorând noua stare (inclusiv legătura inversă spre obiectul curent) și are loc invocarea recursivă x.p(**this**). Apoi se readuce mat[i][j] la valoarea 0;
- mat[i][j]=-1 : nu se poate avansa, deoarece s-ar închide un ciclu.

Metoda principală din clasa DrumPlan începe prin inițializarea câmpurilor statice din clasa elem; aceasta se realizează prin intermediul obiectului Ob de tipul elem pentru care se folosește constructorul cu nici un parametru. Este creat apoi obiectul corespunzător stării inițiale (folosind cel de al doilea constructor) și pentru acest obiect este invocată metoda back cu parametrul **null**, corespunzător faptului că starea inițială nu are un predecesor.

Programul este următorul:

```
import java.io.*; import input;

class elem {
  int i,j; elem prec;
  static int m,n,i0,j0;  static int[][] mat;
  static int ndepl;
  static int[][] depl = { {1,0,-1,0}, {0,-1,0,1} };
  static StreamTokenizer f =
                  new StreamTokenizer(System.in);

  static { ndepl=depl[0].length; }

  elem() {
    int i,j;
```

Menționăm că modificatorii însoțesc nu numai câmpurile, ci și metodele, dar vom amâna discuția privind efectul modificatorilor asupra metodelor. La declararea câmpurilor unei clase se recomandă precizarea modificatorilor în ordinea de mai sus. Modificatorul **static** specifică faptul că respectivul câmp este unic pe clasă, deci comun tuturor obiectelor ce sunt instanțe ale clasei.
Numai câmpurile pot fi declarate cu **final**. Variabila declarată cu acest modificator primește doar o singură dată o valoare și anume la declararea sa. O declarare poate să apară oriunde în textul sursă, deci nu neapărat la începutul unei clase, metode sau al unui bloc de inițializare.
Limbajul Java atribuie valori inițiale prin lipsă câmpurilor claselor. Ele sunt:

false (pentru **boolean**) ;
'u0000' (pentru **char**) ;
null (pentru referințe obiect)
0 (pentru orice tip întreg) ;
+0.0f sau +0.0d (pentru **float** și **double**) .

În schimb nu este atribuită vreo valoare inițială prin lipsă variabilelor locale din metode, constructori și blocuri de inițializare și de aceea aceste variabile trebuie să primească valori înainte de a fi utilizate; nerespectarea acestei reguli conduce la semnalarea unei erori.
O variabilă are un nume (care este un identificator) și un tip. Enumerăm în continuare *tipurile primitive (de bază)*.
1) *Tipul* **boolean**. Variabilele de acest tip sunt reprezentate pe un bit și pot avea doar valorile true și false.
2) *Tipul* **char**. Variabilele de acest tip sunt reprezentate pe 16 biți și pot primi ca valoare orice simbol din codul Unicode. O variabilă de tip caracter poate fi folosită oriunde poate apare o valoare întreagă: este considerat numărul său de ordine în setul de caractere Unicode.
3) *Tipuri întregi*. Ele sunt următoarele: **byte** (octet), **short** (întreg scurt), **int** (întreg) și **long** (întreg lung), reprezentate în ordine pe 1, 2, 4, 8 octeți.
Valorile maxime sunt în ordine: 255, 32767, $\cong 2\times10^9$ și respectiv $\cong 9\times10^{18}$
Precizăm că în calcul tipurile **byte** și **short** sunt convertite la **int**.
4) *Tipurile în virgulă mobilă* **float** și **double**, reprezentate respectiv pe 4 și 8 octeți. Referitor la valorile pe care le pot lua, specificăm că exponenții iau valori în intervalele -149..104, respectiv -1045..1000.
Fiecărui tip primitiv din Java, cu excepția lui **byte** și **short**, îi corespunde o clasa definită în java.lang: Boolean,Character,Integer,Long,Float,Double. în aceste clase apar constante și metode utile. Astfel în ultimele patru apar constantele MIN_VALUE și MAX_VALUE, iar în ultimele două apar și constantele POSITIVE_INFINITY și NEGATIVE_INFINITY, toate având semnificații evidente. Mai menționăm metoda isNaN din clasele Float și Double: metoda primește o valoare de tipul corespunzător și întoarce o valoare booleană, care este true dacă și

```
    System.out.println("m,n = ");
    m = (int) input.read(f); n = (int) input.read(f);
    mat = new int[m][n];
    for (i=1; i<m-1; i++)
      for (j=1; j<n-1; j++)
        mat[i][j] = (int) input.read(f);
    for (i=0; i<n-1; i++) { mat[0][i]=2; mat[m-1][i]=2; }
    for (i=0; i<m; i++) { mat[i][0]=2; mat[i][n-1]=2; }
    System.out.println("i0,j0= ");
    i0 = (int) input.read(f); j0 = (int) input.read(f);
  }

  elem(int i, int j, elem x) {
    this.i=i; this.j=j; prec=x; }

  String print(elem x) {
    if (x==null) return "(" + i + "," + j + ")";
    else return x.print(x.prec) +
                "(" + i+ "," + j + ")";
  }

  void back(elem prec) {
    elem x=null; int ii,jj;
    for (int k=0; k<ndepl; k++) {
      ii = i+depl[0][k]; jj = j+depl[1][k];
      if (mat[ii][jj]==1);
      else if (mat[ii][jj]==2)
                System.out.println(print(prec));
      else if (mat[ii][jj]==0) {
            mat[i][j]=-1; x = new elem(ii,jj,this);
            x.back(this); mat[i][j]=0;
          }
      else ;
    }
  }
}

class DrumPlan {
  public static void main(String[] s) {
    elem Ob = new elem();
    Ob= new elem(elem.i0,elem.j0,null); Ob.back(null);
    input.readln();
  }
}
```

Pentru exemplul considerat, ultimele 4 drumuri vor fi:

(3,4)(2,4)(2,5)(3,5)(4,5)
(3,4)(2,4)(2,5)(1,5)
(3,4)(3,5)(4,5)
(3,4)(3,5)(2,5)(1,5)

O metodă de generare a tuturor (n,k) - combinărilor

Fie date numerele naturale n,k cu n≥k>0. Numim (n,k) - *combinare* un k-uplu $(a_1,\ldots a_k)\in\{1,2,\ldots,n\}^k$ cu $a_1<a_2<\ldots<a_k$. Fie C(n,k) mulțimea lor. Se știe că numărul (n,k) - combinărilor este C_n^k, adică $|C(n,k)|$. Ne propunem în continuare să scriem un program care să producă la ieșire toate (n,k) - combinările.

Evident, nu este vorba de a calcula C_n^k, ci de a produce elementele lui C(n,k); altfel spus, cerințele problemei nu sunt de natură cantitativă, ci calitativă. Totuși, aparent paradoxal, vom pleca de la formulele cantitative:

$$C_n^0 = 1 \qquad (1)$$

$$C_n^k = C_{k-1}^{k-1}+\ldots+C_{n-1}^{k-1} \quad \text{pentru} \quad n>k \qquad (1')$$

Formulele cantitative de mai sus sunt ușor de demonstrat prin inducție și apar în orice culegere de probleme ce acoperă clasa a X-a. Ce nu apare în aceste culegeri, făcând să se creadă că este vorba numai de artificii de calcul, este aspectul lor calitativ. Mai precis, semnificația lui (1') este:

$$C(n,k)=\bigcup_{i=k}^{n} C'(i-1,k-1) \qquad (2)$$

unde C'(i-1,k-1) este mulțimea obținută prin adăugarea elementului i la fiecare dintre vectorii de lungime k-1 din C(i-1,k-1). Vom nota acest lucru prin

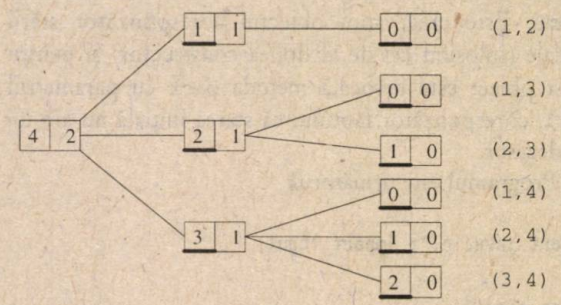
C'(i-1,k-1)=C(i-1,k-1)•i. (3)

Să exemplificăm cele de mai sus pentru n=4, k=2.

C(4,2) = C(1,1)•2 ∪ C(2,1)•3 ∪ C(3,1)•2 =
= {(1)}•2 ∪ {(1),(2)}•3 ∪ {(1),(2),(3)}•4 =
= {(1,2)} ∪ {(1,3),(2,3)} ∪ {(1,4),(2,4),(3,4)} =
= {(1,2),(1,3),(1,4),(2,3),(2,4),(1,4)}

Să mai observă, vezi (3), că elementul care se concatenează lui C(i,j) este i+1, deci valoarea sa este bine determinată și nu mai trebuie "ținută minte" separat.

Se obține astfel următorul arbore și următoarele (4,2) - combinări corespunzătoare frunzelor:



unde am avut grijă ca (4,2) - combinările să apară în ordine lexicografică crescătoare: vezi ordinea membrilor în reuniunea din (2).

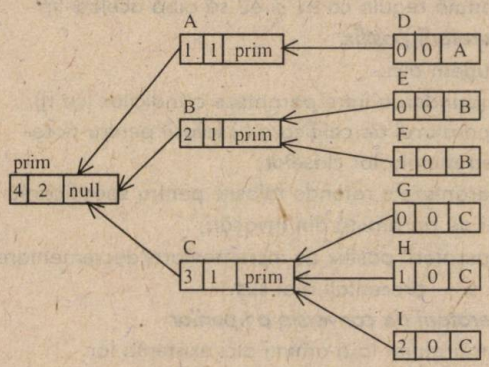
Este clar că:

- dezvoltarea nodurilor din arbore se oprește când k devine 0;
- (4,2) - combinările se regăsesc parcurgând drumul de la frunze către rădăcină.

Plecând de la cele de mai sus, vom construi un arbore

de obiecte pentru care legăturile merg spre tatăl lor. Câmpurile obiectelor vor fi constituite din `n,k` proprii și obiectul precedent.

Dorim deci să obținem următorul arbore de obiecte (în care obiectele au primit nume doar pentru claritate):



Putem trece acum la descrierea programului.

Metoda principală construiește în `prim` rădăcina arborelui și apoi "comandă" obiectului `prim` să execute metoda sa `p` pentru argumentul `null`.

Clasa `elem` cuprinde câmpurile `n,k,prec` (`prec` indică spre tatăl obiectului curent). Constructorul crează un obiect și precizează legătura sa spre tatăl său.

Metoda `p` construiește descendenți ai nodurilor arborelui atâta timp cât `k>0`; dacă s-a ajuns la `k=0`, atunci se invocă metoda `print` pentru parcurgerea drumului de la frunză spre rădăcină și implicit pentru listarea `(n,k)` - combinării asociate.

Să considerăm situația în care obiectul `E` invocă metoda `p` (cu parametrul `B`). Cum `k=0`, obiectul `E` va executa metoda `print` cu parametrul `B`. Ca urmare va fi întors șirul format din `1` concatenat cu rezultatul invocării `B.print(prim)`. Dar acest rezultat este șirul format din `3` urmat de rezultatul invocării `prim.p(null)`, care este șirul vid. În consecință la ieșire va apare combinarea `(1,3)`.

În clasă a mai fost folosit câmpul static `cnk` ce numără, pentru verificare, câte `(n,k)` - combinări au fost generate. În final, metoda principală va tipări și această valoare.

Programul este următorul:

```
import java.io.*; import input;

class elem {
  int n,k; elem prec;
  static int cnk=0;

  elem(int n, int k, elem x) {
     this.n=n; this.k=k; prec=x;}

  String print(elem x) {
    if (x==null) return "";
    else return n+1 + " " + x.print(x.prec);
```

numai dacă argumentul nu este o valoare numerică validă.

Operatori

Operatori aritmetici

Acești operatori sunt: + - * / % .
De asemenea este folosit operatorul unar - pentru schimbarea semnului, precum și operatorul unar + (introdus pentru simetrie).
În aritmetica întreagă trebuie să ținem cont de următoarele reguli:

- orice valoare ce depășește limitele admise este redusă modulo această limită; de aceea nu există depășiri (*overflow* și *underflow*);
- împărțirea întreagă se face prin trunchiere; de exemplu 8/3==2 și -8/3==-2 ;
- operatorul % satisface regula:

(x/y)*y+x%y==x
De exemplu 8%3==2, iar -8%3==-2.

Operatorii de incrementare și decrementare

Este vorba de operatorii ++ și —.
Expresiile x++ și ++x sunt echivalente cu x=x+1, cu excepția faptului că x este evaluat o singură dată. Această excepție are în vedere instrucțiuni de tipul a[indice()]=a[indice()]+1, în care este folosită o metodă ce întoarce un întreg; astfel de exemplu în a[indice()]++ nu va avea loc de două ori invocarea metodei indice.
Diferența între x++ și ++x constă în faptul că incrementarea este realizată după, respectiv înainte, de utilizarea lui x în contextul în care apare. Astfel, dacă valoarea curentă a lui x este 4, atunci:

- evaluarea expresiei 2*++x conduce la rezultatul 10,
- evaluarea expresiei 2*x++ conduce la rezultatul 8,

după care valoarea lui x va fi 5.
Evident, operatorul -- se supune unor reguli analoage.
O construcție de tipurile x++,x—,++x,—x reprezintă o expresie aritmetică, dar și o instrucțiune (valoarea expresiei nu este folosită, dar x este incrementat sau decrementat).
Operatorii de mai sus pot fi aplicați și pentru tipul **char**, semnificând trecerea la precedentul, respectiv următorul caracter din setul de caractere Unicode.

Operatorii de atribuire

Pe lângă operatorul = folosit standard pentru atribuire, mai pot fi folosiți și operatorii: += -= /= *= %= care reprezintă scrieri prescurtate. Astfel x+=y este echivalent cu x=x+y, cu excepția faptului că evaluarea lui x se face o singură dată (vezi paragraful precedent). Desigur, operatorul %= va fi folosit doar pentru aritmetica întreagă.

Operatorii relaționali

Este vorba de următorii operatori:
> (mai mare ca) >= (mai mare sau egal cu) < (mai mic ca)
< (mai mic ca) <= (mai mic sau egal cu) != (diferit de)

Operatori booleeni

Este vorba de următorii operatori:
|| (sau) && (și) ! (negare)
cu mențiunea că la evaluare se face scurtcircuitare.

```
  }

  void p(elem prec) {
    elem x=null; int i;
    if (k==0) {System.out.println(print(prec)); cnk++;}
    else for (i=k; i<=n; i++)                    // (*)
          {x=new elem(i-1,k-1,this); x.p(this); }
  }
}

class Comb {
  public static void main(String[] s) {
    StreamTokenizer f = new StreamTokenizer(System.in);
    int n,k;
    n = (int) input.read(f); k = (int) input.read(f);
    elem prim = new elem(n,k,null); prim.p(null);
    System.out.println("n= "+n+" k= "+k+
                       " cnk= " + elem.cnk);
    input.readln();
  }
}
```

Programul de mai sus, pe lângă eleganță, concizie și inedit mai are următoarele trăsături, mult mai importante:

- este realizată încapsularea câmpuri + metode;
- nu se efectuează operații ca de exemplu regăsirea unui element al vectorului având o anumită proprietate;
- este creat un nou obiect dacă și numai dacă suntem siguri că ajungem la o soluție. Cu alte cuvinte putem spune că s-a folosit un backtracking optimal;
- nu sunt "active" niciodată mai mult de k obiecte;
- mai mult, colectorul de gunoaie folosit de Java ne dă posibilitatea de a nu "elibera" explicit spațiu. Memoria ocupată de obiectele create, dar care nu mai sunt active, poate fi folosită în alte scopuri, de exemplu pentru aplicații care se execută în paralel cu programul nostru;
- instrucțiunea `(*)` din program transpune întocmai relația (2). În acest mod programul este exprimat într-o formă strâns legată de specificarea matematică a algoritmului, tendință evidentă în programarea modernă.

Exerciții:

1) Încercați să implementați programul de mai sus în Pascal sau C;

2) Folosind schema de mai sus, încercați să determinați toate `(n,k)` - combinările folosind formula:

$$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$$

❐

Prof. dr. Horia Georgescu este cadru didactic al Facultății de Matematică a Universității din București. E-mail: hg@math.math.unibuc.ro

Operatorul condițional ?:

Acest operator constă într-o expresie de forma:
(*cond* ? *e1* ; *e2*)
a cărei valoare este *e1* dacă *cond* este true, respectiv *e2* dacă *cond* este false.
Pentru a nu intra în amănunte, vom impune deocamdată regula ca *e1* și *e2* să aibă același tip.

Operatorii postfix

Includem aici:

- cuprinderea între paranteze a indicilor (cu []);
- operatorul de calificare (.) folosit pentru accesarea membrilor claselor;
- parantezele rotunde folosite pentru specificarea listei de parametri din invocări;
- operatorii postfix de incrementare/decrementare ++ și -- prezentați mai sus.

Operatorii de conversie a tipurilor

Ne mărginim la a aminti aici existența lor.

Operatorul + pentru lucrul cu șiruri

Acest operator este folosit pentru concatenarea șirurilor. Trebuie menționat că dacă un membru al unei sume este un șir de caractere, atunci are loc o conversie implicită a celorlalți membri ai sumei (devenită acum concatenare) la șiruri de caractere; această facilitate a fost de altfel des folosită în programele prezentate, de exemplu în cadrul invocării System.out.print(...).

Asociativitate

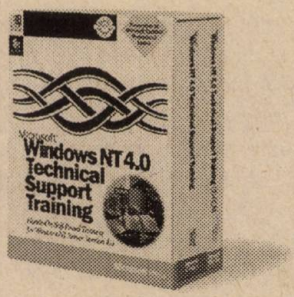
Toți operatorii binari sunt asociativi la stânga.
Operatorul de atribuire este asociativ la dreapta.
Astfel instrucțiunea:
x = y = z = 0;
este echivalentă cu;
x = (y = (z = 0));
și are ca efect atribuirea valorii 0 variablilelor x,y,z.
Explicația constă în aceea că efectuarea unei atribuiri conduce și la obținerea unui rezultat (valoarea care a fost atribuită) și care poate fi folosită, de exemplu ca mai sus.

Precedența operatorilor

Ordinea în care are loc efectuarea prelucrărilor prevăzute de operatori este dată în următorul tabel de priorități ale operatorilor (de la prioritate maximă la prioritate minimă):

Operatorii prefix

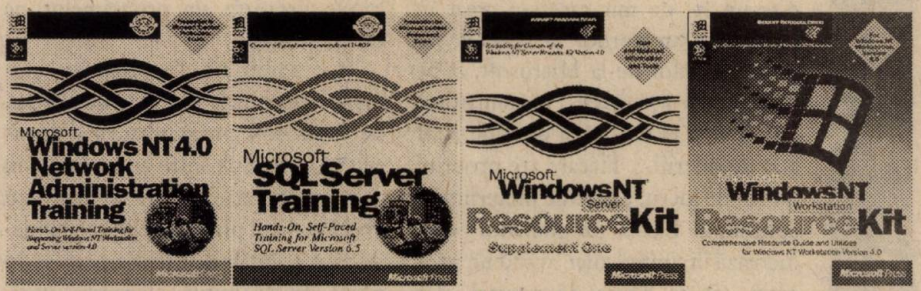
- operatorii unari de: incrementare/decrementare, operatorii + și - unari, operatorul de negare ! și operatorii de conversie de tip
- operatorii multiplicativi: * / %
- operatorii aditivi: + -
- operatorii relaționali: > >= < <=
- operatorii de egalitate: == !=
- conjuncția logică ||
- disjuncția logică &&
- operatorul condițional ?:
- operatorii de asignare.



Microsoft Press știe

este timpul

Îți lipsesc. **Ți le-ai dori.** *Resursele pentru instruire și certificare Microsoft.* **Norocul** *tău,* **Microsoft Press are soluția;** *seturile de resurse și seturile de instruire în ritm propriu care te ajută să* **înveți** *despre ceea ce ai* **nevoie,** *atunci când ai nevoie să știi. Indiferent că dorești să-ți* **perfecționezi** *cunoștințele anterioare sau să-ți creezi un* **program de instruire într-un ritm propriu,** *vei afla că mai ai și* **timp pentru viața** *ta.*



Microsoft® Windows NT 4.0 Network Administration Training
1-57231-439-7

Microsoft® SQL Server Server Training
1-55615-930-7

Microsoft® Windows NT® Server 4.0 Resource Kit
1-57231-344-7

Microsoft® Windows NT® Workstation 4.0 Resource Kit
1-57231-343-9

Microsoft Press

COMPUTER PRESS
AGORA

Cărțile pot fi răsfoite, comandate și cumpărate de la librăria noastră deschisă la sediul firmei **Forte Computers**, *str. Lipscani nr. 102; precum și de la următoarele magazine din* **București**: *Flamingo Computers, Bd.N. Titulescu nr.102; Raffles Computers Shop, Calea Victoriei nr.25; Librăria "Noi", sala Dales; Librăria Humanitas din Pasajul Cretzulescu (lîngă magazinul Muzica); MIVAS: Pasaj Universitate;* **BRAȘOV** *Intreprof International, str. Operetei nr.86;* **TIMIȘOARA:** *Computer 21 SRL Bul. Regele Ferdinand nr. 2.;* **ARAD:** *BB Computer, b-dul Revoluției 26-38*
Puteți comanda orice carte de la: **Computer Press Agora s.r.l.**, *C.P. 94 OP.49, București, fax: 01-3309285*
Important: Lista completă de prețuri poate fi obținută via BBS 065-210780 login: bbs, terminal: ANSI,...

MONICA de la MICROSOFT

Când a început să colaboreze la PC Report și BYTE România, Monica Pietroșanu era încă studentă. Au trecut doar câțiva ani de atunci. Acum este inginer și lucrează la cea mai mare casă de software din lume, acolo unde orice programator visează să ajungă. Voi nu?

Mircea: Imaginea "clasică" a programatorului: un tip care stă nopțile în fața calculatorului, bea cantități uriașe de cafea tare, e aproape obsedat de un "bug" pe care nu-l poate scoate... Nu prea-i place să primească sfaturi, detestă să scrie documentații, este neajutorat în chestiuni administrative etc. În ce măsură se potrivește acest "portret" programatorilor de la Microsoft?

Monica: Cred că există un strop de adevăr în această descriere. Acesta este un fel de mit care s-a creat în jurul programatorilor în general, nu numai al celor de la Microsoft, și pe care aceștia îl cultivă. Programatorul este însă și el un om și oamenii sunt atât de diferiți... Fiecare are propriul lui stil de lucru, propriile preferințe și aptitudini. Există programatori care lucrează în timpul zilei, având un program cât se poate de normal, dar există și alții care au preferințe pentru orele nocturne. De aceea, în campusul Microsoft, la orice oră poți vedea animație, iar noaptea în orice clădire zărești câteva birouri cu ferestre luminate. O particularitate o reprezintă aici perioadele dinaintea lansării unui produs, când majoritatea celor implicați în dezvoltarea produsului respectiv au programul de lucru mult extins. Atunci se poate întâmpla să lucrezi și noaptea.

În ceea ce privește scrierea documentațiilor, poate că programatorilor nu le prea place să o facă, sau consideră aceasta o treabă de importanță redusă, comparativ cu scrierea programelor. Din fericire însă, în orice firmă mare de software, așa cum este Microsoftul, există departamente speciale care se ocupă cu scrierea documentațiilor și formatarea lor, astfel încât programatorii trebuie doar să trimită acestui departament specificațiile detaliate ale programelor pe care le produc.

La capitolul "acceptarea sfaturilor" treburile stau cu totul altfel. La Microsoft programatorii se consultă foarte mult între ei deoarece, în orice problemă pe care o întâlnești există o mare probabilitate ca altcineva să o fi întâlnit înaintea ta și să îți poată da cel puțin o sugestie utilă. Comunicarea ajută la rezolvarea multor probleme, așa că nu este niciodată evitată.

În chestiuni administrative nu se poate spune că programatorii există neajutorați, ci doar neinteresați. Problemele administrative nu fac parte din activitatea lor zilnică și sunt alte persoane care se preocupă ca programatorii să nu fie confruntați cu ele. Dacă ținem cont de faptul că mulți foști programatori au evoluat la Microsoft în lideri de grupuri, manageri de program sau de proiect și că aceste poziții necesită implicarea în chestiuni mai mult sau mai puțin administrative, se poate spune că un programator poate face și așa ceva atunci când este nevoie.

Cafeaua este într-adevăr un atribut al Microsoftului. De la înființarea firmei, cafeaua a fost gratuită și la discreție. Ea a ajutat la ținerea trează a spiritelor în perioadele de lucru intens dinaintea lansărilor de produse. În prezent însă, nu numai cafeaua ci și răcoritoarele, laptele și sucurile naturale sunt prezente în imensele frigidere de la fiecare etaj al clădirilor din campusul Microsoft, fiind gratuite. Așa că imaginea "programatorului care bea cantități imense de cafea tare" este adeseori înlocuită cu cea a programatorului care bea Coca Cola sau pur și simplu ceai...

Mircea: Toată lumea își imaginează că Microsoft se compune mai ales din programatori. Care este de fapt adevărul? Care este ponderea programatorilor? Care este "greutatea" lor?

Monica: Ponderea exactă a programatorilor este mult mai mică. Mai puțin de 40% dintre angajații Microsoft-ului lucrează în cercetare și

Fișa biografică

Monica Ene-Pietroșanu s-a născut la Sinaia în 1972. Între 1986 și 1990 a urmat cursurile Liceului Industrial din Sinaia – clasă ce profil de matematică-fizică. În 1995 a absolvit Facultatea de Automatică și Calculatoare a Universității "Politehnica" București, specialitatea Calculatoare. În 1996 a absolvit și cursurile Facultății de Matematică a Universității București.

Dintre foștii ei dascăli poartă o deosebită apreciere fostului ei profesor de matematica, dl. Adrian Ghioca (directorul Liceului Sinaia), căruia îi datorează pasiunea pentru matematică. Un specialist de marcă ce a contribuit la formarea ei profesională este dl. prof. dr. ing. Victor-Valeriu Patriciu (Academia Tehnică Militară), de la care provine "apetitul" pentru domeniul criptografiei. De asemenea, dl. prof. dr. ing. Valeriu Cristea, de la Facultatea de Calculatoare, îndrumătorul tezei de doctorat.

Domeniile de interes: Criptografia și în special criptografia cu chei publice. Un alt domeniu de interes îl reprezintă Sistemele de Operare, în special Unix și, mai nou, Windows NT.

dezvoltarea de produse, aici intrând și programatorii. Un alt procent important la Microsoft îl formează însă angajații din vânzări și suport tehnic, mai mult de 45%.

În cercetarea și dezvoltarea de produse există mai multe tipuri de funcții: *cercetători*, *manageri de produse*, *manageri de program*, *proiectanți*, *cei care dezvoltă software*, *ingineri de localizare* și *ingineri de testare*. Toți aceștia au mai mult sau mai puțin de-a face cu programarea (în general balanța se înclină spre "mai mult" și "foarte mult"). *Cei care dezvoltă software* sunt inginerii care scriu efectiv codul pentru produsele MS. *Inginerii de localizare* se ocupă cu traducerea diferitelor produse ale Microsoft în diferite limbi și localizarea lor pentru piețe din diferite părți ale lumii. *Inginerii de testare* sunt cei care proiectează și implementează softul pentru testarea produselor. O altă categorie o reprezintă *inginerii care testează efectiv produsele* pe diferite platforme hardware și în diferite configurații.

Mircea: Această pondere este specifică MS?

Monica: Cred că ponderea este specifică marilor firme de software, nu Microsoftului. Orice firmă mare care dezvoltă software comercial trebuie să aibă în structura ei o astfel de repartizare care să permită o totală flexibilitate. Pe piața software-ului și hardware-ului apar tot timpul lucruri noi. De aceea o astfel de companie trebuie să aibă în permanență oameni care să fie la curent cu noutățile, să prospecteze piața și să decidă ce proiecte sunt de perspectivă. De asemenea, trebuie să existe oameni care să poată fi implicați rapid în proiecte și, în același timp, oameni care să asigure în permanență coerența și buna realizare a acestor proiecte. Deci există niște reguli generale, născute din necesitate, ce guvernează lumea software-ului comercial pe care, în general, toate marile firme le adoptă.

Mircea: Poate ar fi interesant să ne spui ce alte slujbe interesante există (în afară de programatori).

Monica: Există o serie de alte slujbe interesante și m-aș opri aici asupra celei de *manager de program* și celei de *manager de produs*. Managerii de produse au un rol foarte important în lansarea și susținerea produselor pe piață, dar tot ei decid și ce noi facilități vor fi adăugate sau extinse în versiunile următoare ale unui produs. Ei trebuie să fie tot timpul în contact cu specialiștii care studiază piața, să cunoască foarte bine detaliile tehnice ale produsului care îl promovează, să participe la realizarea lui din momentul proiectării și elaborării specificațiilor, până la faza de întreținere și extindere. Managerii de program se ocupă în general de respectarea specificațiilor produselor și adaptarea lor pe parcursul dezvoltării efective a produselor.

Alte "variațiuni" interesante de programatori sunt *inginerii de localizare*. Cine nu a avut de-a face cu aceasta nu își poate imagina câte lucruri trebuie luate în considerare la traducerea și localizarea unui produs. De la adaptarea tuturor elementelor interfeței (dimensiunile cuvintelor diferă total în germană, de exemplu, de cele în engleză), la cele mai bizare facilități care trebuie adaptate la caracteristicile țării pentru care este tradus produsul respectiv.

Un job deosebit de interesant îl au și inginerii de la "uzabilitate" (*Usability Engineers*) care studiază gradul de uzabilitate al fiecărui produs și modalitățile de îmbunătățire a acestuia.

Sunt multe slujbe, fiecare interesantă în felul ei, care pot să fie importante într-o astfel de companie și toate necesită oameni care să-și cunoască bine treaba și să o realizeze cu promptitudine și rigurozitate.

Mircea: Cum se desfășoară o zi obișnuită din viața ta la Microsoft?

Monica: La prima oră (ca orice

angajat al Microsoftului care se respectă :-)) îmi citesc poșta electronică. Toate rezultatele testelor de laborator rulate peste noapte sunt comunicate pe această cale. Oricum, activitatea de citire a poștei electronice se repetă în cursul zilei, deoarece apar tot timpul noi probleme legate de versiunea curentă aflată în lucru care fie trebuie să primească răspuns, fie trebuie doar să fii la curent cu ele, fiindcă au implicații asupra modulului la care lucrezi tu.

Apoi se poate spune că începe ziua de lucru obișnuită în care fie proiectez ce va urma să fac, fie scriu cod, fie rezolv bug-uri, fie rulez teste în laborator. Depinde de perioadă, dacă produsul la care lucrezi este încă departe de data lansării, se poate ca o mare parte din timp să o petreci încercând să concepi cât mai eficient codul și să îl implementezi în mod optim. Dacă se apropie data lansării, atunci o mare parte din timp o petreci fixând bug-uri și rulând teste. Uneori poți pierde ore cufundat în cărți și căutând prin help-uri soluția la o problemă de care te lovești.

În caz de nevoie, atunci când nu găsești de unul singur răspunsul de care ai nevoie, dai o raită pe coridor, prin birourile colegilor de echipă și, după discuții mai mult sau mai puțin aprinse, te întorci cu o soluție.

Aici peste tot, în birouri, pe coridoare și în sălile de ședință, există table albe pe care programatorii le umplu de desene care mai de care mai enigmatice, în cursul unor astfel de discuții. Ilustrarea ideilor este un factor deosebit de important aici pentru găsirea răspunsului corect.

De câteva ori pe săptămână se țin, în cursul dimineții sau după-amiezii, ședințe la nivel de grup de lucru sau cu toată echipa, în cadrul cărora se discută stadiul actual al proiectului, noile orientări sau pur și simplu se încearcă rezolvarea în comun a unei probleme. Comunicarea este foarte importantă. Uneori, în colectiv găsești într-un sfert de oră răspunsul pe care, singur îl poți căuta ore sau chiar zile întregi. Și reciproc este adevărat, o discuție care îl poate ajuta pe un coleg de echipă, o accepți întotdeauna cu plăcere și promptitudine.

Prânzul este, în general, marcat de o ieșire în grup la cantina din clădirea respectivă. Este o mică recreație pe care majoritatea programatorilor și-o oferă în timpul zilei. Acolo poți întâlni colegi din alte echipe sau co-naționali. În general acolo se întâlnesc foarte des românii, dat fiind că există o mare aglomerație de români în clădirile departamentului "Windows NT".

După prânz, revii la problemele din birou.

Adeseori zilele sunt marcate de diferite evenimente ce au loc în campus: prezentări de produse noi (făcute

de Microsoft sau alte firme), diferite sărbătoriri haioase, cursuri pe anumite teme sau workshop-uri.

Programul unei zile de lucru aici este întotdeauna plin de neprevăzut și deosebit de colorat. Eu sunt de puțină vreme aici, dar șefa mea spunea că lucrează de trei ani la Microsoft și caracteristica principală aici este aceea că nu te plictisești niciodată. Am început deja să mă conving de acest adevăr.

Mircea: Sunt mulți străini? Dar români?

Monica: Sunt extrem de mulți străini. Campusul Microsoftului este cel mai colorat loc din regiunea aceasta, cred. În grupul meu de lucru (care numără cinci oameni) exista un brazilian, un indian, o româncă (eu) iar ceilalți doi sunt, totuși, americani. Ca să nu mai spun că în echipa mare sunt indieni, chinezi, francezi, danezi, nemți, cehi și s-ar putea să-mi fi scăpat totuși câteva naționalități. Această diversitate de culori naționale face ca oamenii să fie destul de îngăduitori și de înțelegători unii cu ceilalți, iar noii veniți să se integreze destul de repede. Limba care se folosește este, evident, engleza dar există atâtea accente și dialecte...

Numărul românilor la Microsoft a început să crească vertiginos și acesta este un motiv de bucurie pentru mine. Românii sunt, în general, foarte bine pregătiți, cursurile care le fac în facultate și în școală le dau un bagaj de cunoștințe simțitor mai mare decât al studenților americani. Așa că românii se prezintă foarte bine la interviurile pe care Microsoftul le organizează anual la București și reușesc de multe ori să fie angajați la această firmă. Legat de acest subiect, aș vrea să spun că, din momentul în care am ajuns în Statele Unite, respectul meu pentru studenții și profesorii români a crescut extrem de mult când am văzut diferența dintre resursele de pregătire pe care le au la dispoziție studenții americani și studenții români. Deși școlile și universitățile din România au început în ultimii ani să fie bine dotate din punct de vedere al tehnicii de calcul și al resurselor de informare, totuși nu se poate spune că accesul la informația tehnică, legată de calculatoare, pentru români este la fel de facil ca pentru americani. O simplă intrare într-o librărie din SUA este suficientă pentru a te convinge de acest adevăr. Cărți de calculatoare se aduc și în România, dar câți își permit (deocamdată) să le cumpere? Pe când aici ai acces la absolut tot ce a apărut nou în domeniu și ești în permanent contact

cu aceste noutăți. Ei bine, chiar și în aceste condiții, studenții români (și mă refer în primul rând la studenții din domeniul calculatoarelor, cu care am avut un contact mai strâns înainte de plecarea mea din țară) sunt **extrem** de bine pregătiți. Toată admirația mea și pentru profesorii români, care actualizează în permanență programa de studiu și conținutul cursurilor lor în concordanță cu ultimele subiecte de interes în domeniu, astfel că, la ora actuală, un absolvent de Calculatoare de la noi are un bagaj complet de cunoștințe și face față cu succes unor selecții extrem de pretențioase, așa cum este și cea a Microsoftului. În România se pregătesc specialiști de înaltă clasă și acest lucru cred că nu trebuie uitat, oriunde te-ai afla în lume.

Mircea: Care sunt relațiile inter-umane într-o comunitate atât de diversă din punct de vedere etnic și cultural?

Monica: În comunitatea românilor în particular, relațiile inter-umane sunt bune, zic eu. Aproape toți românii care lucrează la ora actuală aici au trecut prin aceleași experiență la început, se izbesc încă de o serie de aspecte cotidiene comune și schimbul de experiența în aceste situații este extrem de util, mai ales când vorbești aceeași limbă și vii de pe aceleași meleaguri. În plus, mulți au fost colegi de facultate sau de serviciu în România și rămân în continuare foarte apropiați și aici. Românii se ajută foarte mult între ei și este foarte bine că atunci când vii pentru prima dată aici ai pe cine să întrebi cum și ce să faci în problemele de zi cu zi, care sunt destul de diferite de cele din țară.

Pe de altă parte, relațiile inter-umane în comunitatea multicoloră a microsoftiștilor sunt în mare parte bazate pe pasiunea pentru calculatoare. Lucrul în echipă, respectul și înțelegerea reciprocă sunt foarte importante într-o astfel de firmă. Faptul că majoritatea celor de aici sunt tineri, entuziaști și pasionați de ceea ce fac, contribuie la stabilirea unor relații inter-umane și profesionale foarte strânse. Microsoft-ul face foarte mult pentru întărirea relațiilor dintre angajați prin organizarea de diferite întâlniri, reuniuni, serate și prin atmosfera de entuziasm și comunicare ce există și este întreținută aici. Poți vedea la tot pasul dorința de a ajuta pe fiecare individ să se integreze cât mai bine în această comunitate imensă. Unul dintre vicepreședinții companiei ne-a spus în prima zi de lucru, la festivitatea organizată pentru fiecare grup nou de angajați, că oamenii sunt bogăția cea mai de preț a Microsoft-ului. Nu calculatoarele, nu clădirile, ci calitatea oamenilor care lucrează aici contează și ne-a asigurat că vom fi tratați ca atare. Într-adevăr, așa a fost, nimic până acum nu a dezmințit acest lucru.

Poate părea destul de utopic portretul Microsoftului, așa cum am încercat să-l schițez, dar este cât se poate de realist, toți cei care lucrează aici împărtășesc entuziasmul meu și poate că acei cititori ce îmi vor călca pe urme (eu le doresc succes tuturor celor ce vor încerca) se vor convinge de adevărul celor prezentate mai sus. Părerea mea este că experiența pe care o capeți lucrând într-o astfel de firmă gigant este inegalabilă. Sunt foarte multe lucruri de învățat aici, atât în plan profesional cât și uman. Această experiență va fi cu siguranță extrem de utilă celor ce lucrează aici după ce se întorc în România.

A ajunge aici nu este un noroc sau un vis inaccesibil, ci un rezultat la care orice absolvent român, cu un bagaj solid de cunoștințe și cu pasiune pentru ceea ce face, poate să-l obțină. Numărul românilor de aici care crește continuu este o dovadă clară în acest sens.

Mircea: Ar mai fi clasica pseudo-întrebare de final: ce-ai fi vrut să te întreb și nu te-am întrebat? Și eventual un răspuns...

Monica: Ce nu m-ai întrebat și poate aș fi vrut să mă întrebi (deși am mai discutat noi pe aceasta temă) este dacă un specialist român, ajuns să lucreze la o mare firmă din Statele Unite își mai poate dori să se întoarcă în România sau nici nu mai concepe așa ceva.

Iată și răspunsul, care este o părere extrem de personală. Consider că experiența pe care o capeți lucrând la o firma atât de mare cum este Microsoft și trăind într-o țară cu un nivel tehnic atât de ridicat cum sunt Statele Unite, toate acestea sunt într-adevăr unice. Dar o astfel de experiență capătă cu adevărat valoare doar revenind în România și încercând să dai ceva în țară din ceea ce ai acumulat.

Așa cum am mai spus, în România se pregătesc specialiști de valoare și cred că datoria acestora este să nu uite de unde au plecat și să își aducă aportul lor în țară, căci acolo este cu adevărat nevoie de așa ceva.

La ora actuală se înregistrează un adevărat exod peste hotare al specialiștilor români, în special al celor tineri. Aceasta este rău pentru țară și se datorează condițiilor de acolo. Poate fi totuși un lucru bun, dacă măcar o parte din cei plecați se vor întoarce și vor aduce cu ei experiențe valoroase. Eu una împărtășesc ideea întoarcerii în țară peste câțiva ani și sunt mulți români aici care au aceeași părere.

Cred că dezvoltarea domeniului calculatoarelor este inerentă în România și că nivelul școlii românești va face ca întotdeauna să existe un mare potențial uman și intelectual acolo. În aceste condiții, orice specialist întors din străinătate va găsi un mediu de lucru adecvat cu aspirațiile și pregătirea lui. Dotările din universități se îmbunătățesc în continuu, chiar dacă uneori într-un ritm prea lent, dar progresul se simte clar. Există firme mari de calculatoare care se extind tot mai mult și în afara Bucureștiului. Există reviste de informatică extrem de bune, edituri specializate în cărți de calculatoare, toate având la bază oameni competenți și pasionați. Toate acestea creează un mediu intelectual deosebit.

Deci dincolo de greutățile economice actuale, eu sunt optimistă în ceea ce privește dezvoltarea acestui domeniu în România și sunt sigură că specialiștii români își vor găsi locul care îl merita în țara lor.

Mircea Sârbu

CUM s-a născut JAVA

de Gabriel Proșcanu

De data aceasta va fi vorba de un "portret de grup": echipa care a creat un limbaj și o tehnologie care are mari șanse să revoluționeze (pentru a câta oară?) informatica. De ce este "ciudată" această poveste? Pentru că de fapt eroii ei căutau cu totul altceva...

Totul a început în ziua de 5 decembrie 1990. La vremea aceea, eroul principal avea doar 25 de ani...

M-am săturat!

"Îmi fac bagajele și mă duc la NeXT". Aceasta au fost primele cuvinte pe care Patrick Naughton, inginer software la Sun Microsystems, le-a spus intrând în biroul prietenului său Scott McNealy. Scott era președintele unei ditamai companii și știa că lucrurile trebuie luate cu mult calm. "Ce s-a întâmplat?" l-a întrebat, făcându-i semn să ia loc. "Ce ți-a venit, așa, deodată?" Patrick a mormăit ceva despre haos și despre faptul că se săturase să încerce să lucreze cu sutele de combinații diferite ale API folosite de toți programatorii de la Sun.

Nu e plăcut să auzi că este haos în compania pe care o conduci; dar să ți se ofere această concluzie și să o treci cu vederea, nu este înțelept. Scott l-a ascultat cu atenție pe colegul său din echipa de hochei, apoi l-a surprins pe acesta, dându-i dreptate. "Dacă într-adevăr vrei să pleci, probabil că te-ai gândit bine la pasul ăsta pe care vrei să-l faci. Eu nu te pot împiedica, deși mi-ar părea rău să ne despărțim. Oricum, fie că te vei răzgândi și vei rămîne, fie că vei pleca, fă-mi o ultimă favoare: pune pe hârtie tot ce mi-ai spus mie aici".

Lui Naughton, care era pregătit să aibă o conversație tensionată, acest preț minim pe care îl plătea pentru recăpătarea libertății sale i s-a părut foarte acceptabil. A întocmit, pe nerăsuflate, o listă cu toate nemulțumirile sale, fără să se mai ferească de ceva. A spus lucruri neplăcute: că arhitectura NeWS, la care se lucra, ar trebui regândită de la zero și că mulți dintre cei peste o sută de programatori care lucrau pentru Window Systems Group nu și-ar mai găsi locul, dacă Sun ar face ordine în serviciul său tehnic.

Naughton a depus memoriul respectiv și și-a luat gândul de la întreaga chestiune. A început să-și strângă lucrurile de prin birouri și să-și piardă vremea citind tot felul de reviste, în așteptarea aprobării plecării sale.

Omul ăsta are dreptate!

Peste câteva zile, în locul mesajului administrativ pe care îl aștepta, au început să-i vină cu totul alte mesaje. "Bravo! Ai spus ceea ce mulți nu aveau curajul să spună!", "Este foarte adevărat și te susțin întrutotul", "Cred că a venit vremea schimbării"... Ce se întâmplase? Scrisoarea sa fusese trimisă, în mod confidențial, pe la toți șefii echipelor de programatori. Toate aceste manifestări de solidaritate îl făceau să zâmbească pe Patrick Naughton, dar într-o dimineață, pe când își bea cafeaua, frunzărind mesajele primite, nu a mai reușit să ducă până la gură cana sa mov, zărind o ofertă semnată de Bill Joy (unul dintre fondatorii Sun), James Gosling și alți membri din consiliul de administrație.

Tot ce îți poftește inima

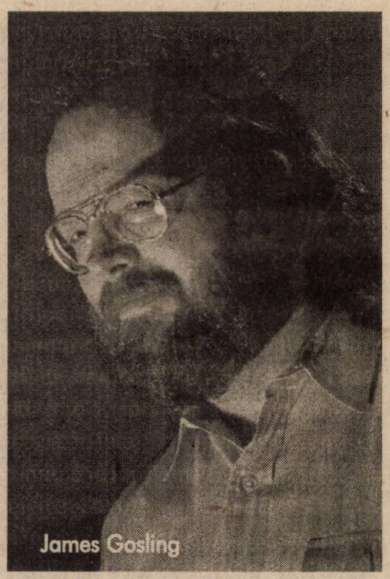
"Îți alegi programatorii pe care îi dorești", "Nu vei avea nici o restricție", "Îi vei organiza cum crezi de cuviință", "Dezvoltați ce anume hotărăști tu"...

Milioane de programatori din lumea întreagă visează o asemenea ofertă. Lui Naughton nu-i venea să creadă că așa ceva chiar ar putea să existe. Câteva clipe a stat amuțit, neputând să-și revină din uimire. Când a reușit, cu o mână tremurândă, să-și ducă la gură cana cu cafea, s-a temut că i-a scăpat ceva. Oare chiar nu era și vreo cerință specială în această ofertă incredibilă?

Evident, oferta conținea și o cerință. Aceasta era la fel de incredibilă: "Dezvoltă ceva *cool*".

Cum totul părea ireal, Patrick a

Patrick Naughton


James Gosling

forțat nota. A cerut ca proiectul să fie secret și să nu i se impună compatibilitatea cu platformele Sun existente. Și un buget de un milion de dolari în primul an. Totul a fost acceptat!

The A-Team

Echipa a fost alcătuită din doar șase oameni, și-a luat numele de cod "Green" și, după modelul savanților care au creat bomba atomică, și-au autoimpus un fel de exil. Pentru a-l face mai suportabil, și-au burdușit un frigider uriaș cu Coca Cola și Dove. Au luat cu ei o grămadă de aparate electronice, de la televizoare și telecomenzi, la Nintendo Game Boys și au început să discute despre ce le plăcea și ce nu le plăcea la aceste aparate.

Nintendo și telecomenzi?

Ce căutau aceste unelte casnice într-o discuție a unor programatori de elită? Simplu: cu ajutorul lor se studia felul în care se pot transmite mesaje între lumi diferite. Un video sau o combină Hi-Fi sunt construite în jurul unor procesoare diferite, iar dacă producătorul lor vrea să le adauge noi funcții este limitat de partea hardware. În plus, cipurile aveau seturi de instrucțiuni foarte limitate.

În mod evident, se impunea o nouă gândire a programării, cu năzuințe de universalitate.

Ce ar vrea să însemne aceste vorbe mari?

Această nouă gândire nu se putea exprima mai bine decât prin crearea unui nou limbaj de programare, orientat pe obiecte, limbaj pe care Gosling l-a botezat *Oak* (stejar) după arborele care era la fereastra încăperii. Bazat întrucâtva pe C++, limbajul nu păstra din acesta decât minimul necesar, pentru a fi compatibil cu spațiul limitat disponibil în cipurile existente în aparatele domestice pe care le studiaseră și pentru a permite programatorilor să lucreze cu un suport hardware foarte variat.

Jocurile sunt de bază...

Pe măsură ce Oak căpăta formă, echipa continua să studieze, foarte amănunțit, motivele pentru care lumea era atrasă de anumite jocuri video și de felul în care au loc interacțiunile cu echipamentele electronice comune. După un timp, au creat un fel de telecomandă, numită "*7", care avea un mic ecran pe care apărea "Duke", un personaj animat, care îi ghida pe utilizatorii care foloseau interfața sa grafică. Ideea de bază a lui *7 era aceea că interfața trebuie să fie atrăgătoare și prietenoasă.

"Duke," creat de Joe Palrang, va deveni, ulterior, mascota Java.

...dar afacerile sunt afaceri

Sun a transformat echipa Green într-o companie numită First Person, a cărei proprietate o deținea. Situația era, totuși, bizară: noua companie se baza pe un concept interesant, dar nu avea nimic de făcut cu acesta... După câteva dezbateri din care s-au străduit să scoată o idee comercială, în final au optat pentru proaspăt apăruta piață a televiziunii interactive.

În lumea televiziunii, însă, lucrurile se mișcă greu. Deși ideea unei televiziuni interactive le părea atractivă, cei de la Time-Warner nu fost încântați atunci când, într-un târziu, și-au dat seama că pentru o asemenea televiziune specială ar trebui și niște televizoare speciale. "Poate după 2010", au spus și s-au răzgândit în ultimul moment, considerând că nu are rost să se aventureze pe un tărâm necunoscut, mai ales că aveau profituri atât de mari din showbiz în forma sa clasică.

Un început dificil

De alt potențial contract cu firma 3DO s-a ales praful, atunci când aceștia au cerut drepturi exclusive asupra tehnologiei.

În acest fel, încercarea de intrare a companiei First Person pe piața televiziunii interactive s-a soldat cu un eșec.

Situația aceasta, deloc roz, avea să se schimbe în 1993, când NCSA (*National Center for Supercomputing Applications*) a introdus Mosaic și a făcut să apară World Wide Web. Foarte repede s-au dezvoltat și alt tehnologii, iar Internetul, odinioară accesibil numai specialiștilor în informatică și universitarilor, a devenit un lucru de folosință comună.

The Internet!

La începutul lui 1994, echipa de la First Person s-a gândit să se concentreze pe aplicații pentru multimedia online. Bill Joy a dus mai departe conceptul inițial, denumind Oak drept "*language based operating system*" și a aderat la ideea lui Naughton de a face publice sursele pe Internet. Astfel, limbajul Oak a devenit el însuși un produs, în loc de a fi un simplu component al vreunui aparat. Arthur van

Hoff a dezvoltat un compilator de Oak, scris 100% în Oak, în loc de tradiționalul C. Naughton și Jonathan Payne au creat un browser cu facilități Oak, denumit "WebRunner." În felul acesta, apărea și primul applet: Duke făcând cu mâna creatorilor săi...

Sun a fost de acord să facă publice sursele limbajului, dar l-a redenumit Java. Acest nume a fost preferat celorlalte care au fost propuse - Neon, Lyric, Pepper și Silk... Povestea spune că era numele sortimentului de cafea preferat de cei de la First Person.

Din moment ce Java ajunsese în Internet, singurul lucru care mai era necesar era o modalitate de a rula applet-uri Java. Datorită unui conflict cu o marcă înregistrată, "WebRunner" a fost redenumit HotJava. La puțin timp după aceea, Netscape a început să suporte Java, deschizând larg o ușă către noua lume...

De fapt, ce anume este Java?

Primul răspuns care ne vine în minte la întrebarea de mai sus este "un mod de a face paginile Web mai arătoase — cu sunete, animații și secvențe video". În realitate, Java este mult mai mult: o platformă computațională, baza pe care se pot clădi aplicații. Proiectanții de software pot crea orice aplicație în Java, de la procesoare de text, la aplicații vitale (*mission critical*), cum ar fi cele de contabilitate, baze de date etc.

Aplicațiile Java (applets) diferă față de aplicațiile obișnuite prin faptul că rezidă în serverele rețelelor. Prin rețea se furnizează applet-ul sistemului care îl cere. De exemplu, pentru a vă verifica situația financiară, vă conectați prin modem (*dial-up*) la serverul băncii unde vă păstrați economiile și vă logați cu ajutorul browserului. Vi se vor transmite datele necesare și un applet pentru a le vizualiza. Dacă doriți să transferați bani dintr-un cont în altul, nu trebuie să vă obosiți cu *cut and paste*. Sistemul băncii vă va furniza un

Cronologie...

(așa cum a fost schițată de Patrick Naughton)

5 decembrie 1990 - Patrick Naughton renunță la decizia sa de a părăsi firma Sun pentru a se alătura firmei NeXT, condusă de Steve Jobs.

15 ianuarie 1991 - Bill Joy, Andy Bechtolsheim, Wayne Rosing, Mike Sheridan, James Gosling și Patrick Naughton participă la un "brainstorming" la Aspen, Colorado. Se caută idei noi pentru așa-numitul "Proiect Stealth".

1 februarie 1991 - Gosling, Sheridan și Naughton încep lucrul. Naughton se concentrează asupra sistemului grafic "Aspen", Gosling asupra limbajelor de programare iar Sheridan asupra aspectelor comerciale.

8 aprilie 1991 - Echipa se mută de la sediul Sun și se deconectează de la rețeaua locală a firmei. Proiectul capătă numele "Green".

15 aprilie 1991 - Ed Frank (arhitectul calculatorului SPARCstation 10), Craig Forrest (proiectantul cipului SS10) și Chris Warth (proiectantul softului NeWS) se alătură echipei Green.

Mai 1991 - Ed Frank născocește numele prototipului hardware: * 7 (se pronunță "star-seven"). De fapt, acesta era codul ce trebuia format la telefon pentru a răspunde la orice apel, de la oricare telefon din firmă.

Iunie 1991 - Gosling începe să lucreze la interpretorul "Oak", care, după câțiva ani, își va schimba numele în "Java".

1 august 1991 - Oak și Aspen sunt combinate și se rulează primul program real.

19 august 1991 - Echipa "Green" prezintă fondatorilor firmei Sun, Scott McNealy și Bill Joy, ideile de bază privind interfața cu utilizatorul și sistemul grafic.

17 octombrie 1991 - Sheridan și Naughton se opresc asupra denumirii "First Person", care urmează să desemneze ideile echipei și, eventual, să fie numele unei noi firme.

17 noiembrie 1991 - "Green" se reconectează la rețeaua internă Sun, printr-o linie de 56K.

1 martie 1992 - Jonathan Payne, cel care mai târziu va conlucra la dezvoltarea programului HotJava, se alătură echipei.

Vara lui 1992 - Se lucrează intens la "Oak", la sistemul de operare "Green", la interfața grafică, la "Star7" și celelalte componente.

4 septembrie 1992 - Dispozitivul "Star7" este gata și este prezentat lui Joy și lui McNealy.

1 octombrie 1992 - Wayne Rosing de la SunLabs este cooptat și preia conducerea echipei.

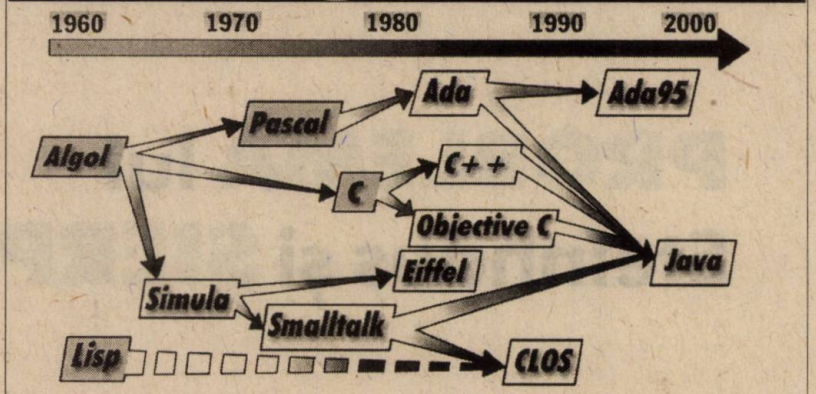
1 noiembrie 1992 - FirstPerson devine firmă.

applet care vă va permite să realizați operațiile dorite.

Din punctul de vedere al unei corporații, Java simplifică crearea și dezvoltarea aplicațiilor, deci este o metodă mai economică. Aplicațiile dezvoltate în Java pot fi rulate, fără vreo muncă de portare, pe orice platformă, decî fără costurile necesare dezvoltării software pentru platforme multiple. În plus, deoarece aplicațiile sunt păstrate doar pe serverele centrale, dispare necesitatea de upgrade cu dischete sau CD-uri, la fiecare computer din rețea.

Ce va aduce viitorul? Nimeni nu poate ști, dar un singur lucru este sigur: nu va mai fi vreodată nevoie să se plângă cineva, așa cum a făcut-o Patrick Naughton, despre incompatibilități la nivel de API...

Genealogie Java



Gabriel Proșcanu este redactor permanent al revistelor PC Report și Byte Romania. E-mail: gproscanu@agora.ro

Cronologie...

15 martie 1993 - FirstPerson se concentrează asupra televiziunii interactive, ca urmare a veștii că Time Warner experimentează un sistem de TV-cablu interactiv în Orlando, Florida.

Aprilie 1993 - Este realizat NCSA Mosaic 1.0, primul browser grafic pentru Internet.

14 iunie 1993 - Time Warner alege Silicon Graphics pentru sistemul său de televiziune interactivă, în ciuda faptului că a recunoscut superioritatea tehnologiei Sun și a asigurat conducerea Sun că îi va încredința contractul.

Vara 1993 - Naughton călătorește în toată America încercând să vândă Oak firmelor specializate în electronică de consum și televiziune interactivă. Este totodată perioada în care popularitatea Internetului crește spectaculos.

August 1993 - Negocieri cu firma 3DO pentru livrarea de componente TV inteligente. 3DO insistă să cumpere tehnologia aferentă. McNealy refuză și tratativele eșuează.

Septembrie 1993 - Un nou proiectant se alătură trupei: Arthur Van Hoff. Deși cooptat pentru dezvoltarea de aplicații pentru televiziunea interactivă, Van Hoff se va ocupa mai mult de proiectarea limbajului.

17 februarie 1994 - FirstPerson prezintă conducerii Sun un plan de afaceri în care accentul este pus pe dezvoltarea de aplicații multimedia, on-line sau pe CD-ROM, pe baza tehnologiei Oak. Planul stârnește controverse.

25 aprilie 1994 - la naștere Sun Interactive. Jumătate din oamenii de la FirstPerson sunt preluați de noua firmă.

Iunie 1994 - Din inițiativa lui Bill Joy, este lansat proiectul "Liveoak", ca o tentativă de a folosi Oak într-un proiect de sistem de operare.

Iulie 1994 - După ce Naughton scrie într-un singur weekend o implementare în Oak a unui browser Web, hotărăște că proiectul "Liveoak" trebuie reorientat spre Internet.

16 septembrie 1994 - Payne și Naughton încep să scrie "WebRunner", un browser asemănător lui Mosaic. Mai târziu, acesta va fi rebotezat "HotJava".

29 septembrie 1994 - Prototipul lui "HotJava" este prezentat conducerii executive de la Sun.

11 octombrie 1994 - Naughton pleacă la firma Starwave, unde este director executiv și vicepreședinte cu tehnologia.

Toamna lui 1994 - Van Hoff implementează un compilator Java în Java (implementarea lui Gosling era scrisă în C).

23 mai 1995 - Sun anunță în mod oficial Java și HotJava la SunWorld '95.

PROBLEMA lui Steinhaus și SIERPINSKI

Ilie Vieru

Enunț: Fiind dat un număr natural m, să se determine în planul euclidian R^2 un cerc care să aibă în interior exact m puncte cu coordonate întregi.

Problema existenței cercului cu proprietatea de mai sus a fost pusă în 1957 de matematicianul ***Hugo Steinhaus***, iar răspunsul afirmativ a fost dat de ***W. Sierpinski*** în 1964 ca o consecință a următoarei observații:

Distanțele de la punctul $w=(\sqrt{2},1/3)$ *la toate punctele lui* $Z*Z$ *sunt diferite între ele.*

Vom demonstra această observație prin metoda reducerii la absurd. Să presupunem că $a=(a_1,a_2)$ și $b=(b_1,b_2)$ sunt două puncte distincte din $Z*Z$ care se află la aceeași distanță de $w=(\sqrt{2},1/3)$. Presupunerea noastră înseamnă:

$$(a_1-\sqrt{2})^2+(a_2-1/3)^2=(b_1-\sqrt{2})^2+(b_2-1/3)^2,$$

adică:

$$a_1^2+a_2^2-b_1^2-b_2^2-a_2*2/3+b_2*2/3-2\sqrt{2}*(a_1-b_1)=0.$$

Avem deci:

$$\begin{cases} a_1-b_1=0 \\ a_2^2-b_2^2-2/3(a_2-b_2)=0 \end{cases}$$

$$\text{adică} \begin{cases} a_1=b_1 \\ (a_2-b_2)(a_2+b_2-2/3)=0 \end{cases}$$

Deoarece $a_2,b_2\in Z$, avem:

$a_2+b_2-2/3\neq 0$ și deci $a_2=b_2$, adică $a=b$, ceea ce contrazice ipoteza și deci, observația este demonstrată.

Pentru a rezolva problema folosind observația lui Sierpinski, notăm:

$Z^2=Z*Z=\{(m_1,m_2)\mid m_1,m_2\in Z\}$.

Evident, Z^2 este o mulțime numărabilă așa încât putem scrie punctele sale ca un șir:

$Z^2=\{a^1,a^2,a^3,...\}$.

Facem aceasta, aranjând punctele a^r în ordinea crescătoare a distanțelor la punctul w, adică:

$|a^r-w|<|a^{r+1}-w|$, $r=1,2,...$

conține punctele cu coordonate întregi $a^1,a^2,...,a^m$ și nici unul altul.

Pentru implementarea *algoritmului de ordonare* a lui Z^2 parcurgem următoarele etape:

fixăm centrul discului conform observației lui Sierpinski, de exemplu $w=(\sqrt{2},1/3)$;

- stabilim domeniul din care prelucrăm datele;
- pentru fiecare punct $a=(a_1,a_2)$ cu coordonatele întregi din domeniu determinăm distanța:

$$d(a,w)=\sqrt{(a_1-\sqrt{2})^2+(a_2+1/3)^2};$$

- cu punctele astfel determinate construim un arbore binar ordonat, după regula:

inf(i)>inf(j) pentru orice vârf j din subarborele stâng al vârfului i;

inf(i)<inf(j) pentru orice vârf j din subarborele drept al vârfului i;

unde inf(i)= $d(i,w)$, respectiv inf(j)= $d(j,w)$;

- traversăm arborele în inordine și afișăm pentru fiecare vârf rădăcină vizitat coordonatele și distanța față de w.

Încercând să generalizăm problema, avem nevoie de următoarele notații:

- $Q^n=\{(r_1,r_2,...,r_n)\mid \forall r_i\in Q\}$, reprezintă mulțimea punctelor din R^n cu coordonate raționale;
- scriem $b-a$ pentru a nota vectorul $\overrightarrow{ab}$ și $|b-a|$ pentru lungimea lui.

Mulțimea Q^n este numărabilă și densă în R^n; aceasta înseamnă că oricare punct din R^n este un punct limită de elemente din Q^n. Proprietatea lui Sierpinski sugerează următoarea:

Problemă:
Să se determine mulțimea S_n de puncte din R^n, astfel încât fiecare punct din S_n să aibă distanțe diferite la toate punctele lui Q^n, adică:

$S_n=\{x \mid x\in R^n, \mid x-x' \mid \neq \mid x-x'' \mid$, dacă $x',x''\in Q^n\}$

Să considerăm la început această problemă pentru cazul cel mai simplu, adică pentru $n=1$. Avem deci, de determinat mulțimea S_1 de puncte reale care se află la distanțe diferite de toate punctele raționale ale lui R. Dacă x este la distanțe egale de două puncte raționale distincte r_1,r_2, atunci:

$x = (r_1+r_2) / 2,\ r_1 \neq r_2$ (1)

și invers. Acest punct x nu aparține lui S_1. Pentru a obține mulțimea S_1 trebuie să îndepărtăm din R toate punctele de forma (1). Dar, punctul (1) este rațional și de asemenea orice număr rațional este media aritmetică a două numere raționale, de exemplu:

$r = ((r-1) + (r+1)) / 2.$

Am demonstrat astfel că:

$S_1 = R \setminus (\cup \{ r \mid r\in Q \}) = R\setminus Q$

și am stabilit următoarea

Teoremă:
Mulțimea de puncte S_1 de elemente din R care sunt la distanțe diferite de toate punctele raționale din R este identică cu mulțimea punctelor iraționale din R.

Aceasta poate fi chiar o definiție a numerelor iraționale.

Pare normal ca această teoremă să se generalizeze la R^n, cu definiții și raționamente corespunzătoare. Să găsim locul geometric al punctelor $x = (x_i)$ care sunt la distanțe egale de punctele a și b din R^n. Formula distanței arată că acest loc este:

$$\sum_{i=1}^{n}(x_i-a_i)^2 = \sum_{i=1}^{n}(x_i-b_i)^2 \cdot$$

Dezvoltând binoamele și reducând $\sum x_i$ din ambii membri, obținem ecuația locului:

$$\left[x_i-(a_i+b_i)/2\right](b_i-a_i)=0 \cdot$$

El trece prin mijlocul $(a+b)/2=\left[(a_i+b_i)/2\right]$ al segmentului $[a,b]$ și este un plan perpendicular pe vectorul

$$\overrightarrow{ab}=b-a=(b_i-a_i)$$

(se mai numește și *bisectorul perpendicular* al punctelor a și b).

Definiții:
Un plan

$$\prod(A_i): A_1x_1+A_2x_2+....+A_nx_n+A_{n+1}=0,$$

$$(\sum_{i=1}^{n}A_i^2>0)$$

se numește *plan rațional* în R^n dacă coeficienții A_i sunt toți raționali, sau — ceea ce este echivalent — întregi.

Vom spune că $x=(x_1,x_2,...,x_n)$ este un punct rațional al lui R^n, dacă se află într-un plan rațional din R^n.

Vom spune că $x=(x_1,x_2,...,x_n)$ este un punct irațional al lui R^n, dacă o ecuație liniară:

$$A_1x_1+A_2x_2+....+A_nx_n+A_{n+1}=0$$

cu toți coeficienții întregi implică faptul că toți A_i sunt nuli (se mai spune că numerele $x_1,x_2,...,1$ sunt *liniar independente*).

Exemple:

a) $x=(\sqrt{2},1)$ este un *punct rațional* în R^2 pentru că x se află pe dreapta rațională $x_2-1=0$;

b) $x=(\frac{\log 2}{\log 7},\frac{\log 3}{\log 7},\frac{\log 5}{\log 7})$ este un *punct irațional* în R^3, deoarece dacă ar fi un punct rațional în R^3, am avea o relație nebanală

$A_1x_1+A_2x_2+A_3x_3+A_4=0$ cu coeficienți întregi, deci:

$A_1\log 2+A_2\log 3+A_3\log 5+A_4\log 7=0$, sau:

$2^{A_1}*3^{A_2}*5^{A_3}*7^{A_4}=1$, ceea ce este imposibil, cu excepția când toți A_i se anulează;

c) $x=(e,e^2,...,e^n)$ este un punct irațional în R^n deoarece nu există o ecuație nebanală de forma:

$c_0+c_1e+c_2e^2+...+c_ne^n=0$ cu coeficienți întregi.

Notăm cu R_n mulțimea punctelor raționale din R^n și cu I_n cea a punctelor iraționale din R^n. Ținând cont de definițiile anterioare putem preciza că I_n este complementara lui R_n în raport cu mulțimea R^n, adică:

$I_n = R^n \setminus R_n$.

De asemenea, mulțimea punctelor raționale R_n se mai poate exprima:

$$R_n=\bigcup_{A_i\in Z}\prod(A_i),$$ adică R_n este reuniunea tuturor planelor raționale din R^n.

O observație simplă dar foarte importantă (a cărei demonstrație nu o includem în materialul de față) este:

$$\bigcup_{A_i\in Z}\prod(A_i)=\bigcup_{a,b\in Q^n}\prod(a,b),\ a\neq b,$$

adică, orice plan rațional $\prod(A_i)$ este și un plan $\prod(a,b)$ determinat de două puncte distincte din Q^n și invers.

Cu aceste rezultate putem scrie:

$$I_n = R^n - \bigcup_{a,b\in Q^n,\, a\neq b} \prod(a,b).$$

Cum membrul din dreapta descrie mulțimea S_n, am stabilit astfel:

Teorema:
Avem identitatea $S_n=I_n$, adică mulțimea S_n a punctelor care au distanțe diferite la toate punctele lui Q^n este identică cu mulțimea I_n a punctelor iraționale din R^n.

Ținând cont de formula distanței în R^n

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}, \quad \forall a,b\in R^n$$

Algoritmul prezentat în acest articol se poate implementa pentru orice $n \geq 2$. Programul alăturat este o implementare pentru $n = 2$.

Bibliografie:

1. Isac J. Schoenberg, *Priveliști matematice*, Ed. Tehnică, București, 1989.

Dl. Ilie Vieru este profesor la Liceul Teoretic "Gh. Vrânceanu" din Bacău.

Listing ORDONARE.PAS

```
Program  ordonare;
uses  Crt;
type  ref=^nod;
      nod=record
            info1,info2:Integer;
           d:Real;
           legs,legd:ref
          end;
var  infnou:Real;
     rad:ref;
     x0,y0:Real;
     a,i,j:Integer;

procedure  prel(tnod:ref);
begin
  Writeln('
',tnod^.info1:5,tnod^.info2:5,tnod^.d:9:5)
end;

procedure  cauta(var  rad:ref);
begin
  if rad=nil
  then
    begin
      New(rad);
      with rad^ do
      begin
        d:=infnou;
        info1:=i;
        info2:=j;
        legs:=nil;
        legd:=nil
      end
    end
  else
    with rad^ do
    begin
      if infnou<d then cauta(legs)
      else if infnou>d then cauta(legd)
    end
end;

procedure   inordine(tnod:ref);
begin
  if tnod<>nil
  then
    begin
      inordine(tnod^.legs);
      prel(tnod); delay(1000);
      inordine(tnod^.legd);
    end
end;

Begin
  x0:=Sqrt(2); y0:=1/3;
  rad:=nil;
  Writeln('Se construieste arborele:');
  Writeln('Dati coltul patratului din cadranul
I:');
  Readln(a);
  for i:=-a to a do
    for j:=-a to a do
    begin
      infnou:=Sqrt(Sqr(i-x0)+Sqr(j-y0));
      cauta(rad)
    end;
  Writeln;
  inordine(rad);
  Readln
End.
```

NEWS: informații și OPINII

de Dragoș Marian și Mircea Sârbu

Forumurile din Internet adună milioane de oameni din peste o sută de țări. Se discută probleme serioase sau doar se bârfește, se leagă prietenii dar au loc și certuri, se împărtășesc pasiuni comune sau se schimbă rezultate ale cercetărilor de vârf. Comunitățile virtuale există. În fiecare zi se publică în forumurile din Internet o întreagă enciclopedie. Să profităm de ea.

În numărul trecut am vorbit despre poșta electronică și, cu această ocazie, am prezentat pe scurt și așanumitele "liste de discuții" (*mailing lists*). Motivul pentru care acestea au fost prezentate în contextul respectiv este faptul că ele folosesc tehnologia e-mail. Este acum momentul să facem observația că poșta electronică (*e-mail*) este, prin concepție și implementare, o modalitate de comunicare punctuală ("de la punct la punct"), în sensul că leagă în mod direct pe cel care trimite mesajul cu cel care-l primește. În schimb, "listele de discuții" reprezintă o modalitate de comunicare în grup (*many-to-many*).

Deoarece listele de discuții reprezintă doar una dintre modalitățile de comunicare în grup facilitate de rețelele de calculatoare, acest articol își propune să completeze peisajul, prezentându-vă în ansamblu unul dintre cele mai fascinante aspecte ale Rețelei: comunitățile virtuale.

Terminologie și clasificare

Cu toate că această prezentare vrea să evite pedanteriile, mijloacele de comunicare în grup sunt atât de diverse și de eterogene încât nu putem evita stabilirea prealabilă a unor jaloane.

Grup și masă. Comunicarea este în grup dacă orice mesaj al unui participant este accesibil tuturor celorlalți participanți și, în plus, fiecare participant are în egală măsură acces la mediul de comunicare. Prima parte a acestei pseudo-definiții distinge comunicarea în grup de comunicarea "punct la punct", în timp ce a doua parte o distinge de comunicarea în masă. Ca exemplu în Internet, o listă de discuții (cum este "dialog") este comunicare în grup, în timp ce o revistă on-line (sau *e-zine* — de pildă "agora ON line") sau un serviciu de știri (de pildă "Agora News") sunt mai degrabă "mass-media", deoarece nu toți receptorii au posibilitatea directă de a comunica mesaje prin intermediul acestora.

Sincron și asincron. O primă disociere privitoare la comunicarea în grup se referă la timp. Comunicarea este sincronă dacă emisia și recepția au loc simultan. În lumea reală, o conferință sau o masă rotundă sunt exemplele cele mai la îndemână. În rețea, video-conferințele sunt încă limitate tehnologic (depind foarte mult de calitatea și capacitatea mediilor de comunicare), cea mai populară formă de comunicare sincronă în grup rămânând încă *chat*-ul — o modalitate prin care se pot trimite și recepționa în timp real (sincron) mesaje scrise. Există mai multe sisteme care permit chat, cel mai popular fiind IRC (*Internet Relay Chat*), pe care îl vom trata într-un articol viitor.

Forum. Acest termen desemnează în principiu orice formă de comunicare în grup asincronă accesibilă în Rețea — subiectul acestui articol. De asemenea, termenul desemnează ansamblul participanților (membri grupului care comunică). Un forum este analog cu un avizier sau "panou de afișaj" (în engleză *bulletin board*), unde oricine poate afișa mesaje și poate citi mesajele afișate de ceilalți. Fiecare forum are o tematică proprie, care stă la baza constituirii sale (un grup se constituie pe baza unui interes comun). Spre deosebire de un panou de afișaj, care are un caracter strict informativ (de regulă, mesajele se rezumă la oferte, cereri și la anunțuri de interes comun), un forum permite și dezbaterea unor subiecte aparținând tematicii sale. Notă: Pluralul englezesc corect este *forums*, dar în jargonul Rețelei se folosește adesea "*fora*".

Listă de discuții (*mailing list*). Este un forum care folosește ca suport tehnic sistemul de poștă electronică. Practic, un *mailing list* funcționează ca un robot care multiplică fiecare scrisoare ce o primește și o trimite tuturor persoanelor care s-au declarat interesate (prin faptul că s-au abonat) de tematica forumului.

Bulletin Board Sistem (BBS) sau "avizier electronic". În general, este un sistem format dintr-un calculator și softul aferent, care oferă utilizatorilor posibilitatea de a publica (termenul consacrat este "a posta") mesaje care vor fi vizibile tuturor celorlalți utilizatori. Mesajele se adresează unor domenii specifice de interes, corespunzătoare unor tematici stabilite. În prezent termenul este folosit aproape exclusiv pentru a desemna un simplu calculator (poate fi un PC oarecare) dotat cu un soft specializat (de pildă Maximus, Waffle, WildCat) și conectat la rețeaua telefonică. Utilizatorii (sau "clienții") BBS-ului se conectează prin modem de la calculatorul propriu, după care sunt "luați în primire" de softul de BBS, care le oferă o gamă de servicii. Forumurile reprezintă doar unul dintre acestea (de regulă se mai oferă e-mail intern, upload/download etc.).

Usenet (sau **net news** — rețea de știri). Este un imens BBS distribuit (BBS este aici considerat în sensul general). Faptul că este "distribuit" înseamnă că reunește și sincronizează un imens număr de "aviziere electronice" și, implicit, forumurile pe care acestea le găzduiesc. Procesele de reunire și sincronizare sunt transparente pentru utilizatori, astfel încât Usenet este perceput ca o colecție unitară de forumuri, care aici se numesc *newsgroups*. Traducerea acestui termen este problematică, deoarece chiar termenul englezesc este oarecum inexact, perpetuându-se mai degrabă din rațiuni istorice (la origini, se postau cu precădere știri, noutăți din diverse domenii). În acest articol vom spune "grup tematic", "grup de știri", "cercuri de discuții", "cercuri de interes" etc., toți termenii fiind echivalenți. Când nu există posibilități de confuzie, se spune doar "grup".

Cum funcționează?

Nu-i deloc ușor de explicat ce este rețeaua de știri. Există mai multe posibile abordări, dintre care cele mai tentante sunt cea istorică și cea tehnică (descrierea implementării). O altă cale posibilă — pentru care am optat în acest articol — este de a o descrie prin contrast cu listele de discuții.

Listele de discuții prin e-mail pot fi considerate un prototip al comunicării asincrone în grup. De altfel, din punct de vedere istoric, ele preced rețeaua de știri. Limitările listelor de discuții încep însă să devină vizibile atunci când numărul abonaților devine foarte mare. Imaginați-vă o listă cu 10.000 de abonați, în care fiecare abonat postează în medie doar un mesaj pe zi. E ușor de socotit cam câte mesaje se generează (cca. 100 de milioane, zilnic) și ce resurse (trafic și spațiu de stocare) sunt ocupate.

O problemă este că o listă de discuții este **centralizată**, fiind susținută de un singur server, astfel încât tot traficul trece în mod necesar prin acest server. O altă problemă o reprezintă faptul ca distribuția mesajelor este **obligatorie**: fiecare abonat primește — fie că vrea, fie că nu — toate mesajele postate, așa încât același mesaj ajunge să fie transportat și stocat de milioane de ori. În fine, o altă problemă este faptul că o listă de discuții este **închisă**: nu este accesibilă decât abonaților ei (exceptând cazul în care o arhivă este publicată pe Internet într-o formă sau alta).

Rețeaua de știri este, în esență, o alternativă care rezolvă aceste probleme: o rețea judicios aleasă de servere de știri (*news servers*), fiecare dintre acestea găzduind toate forumurile. Rezultă de aici că forumurile sunt **descentralizate** (mai precis: distribuite). Utilizatorii pot posta mesaje (care în rețeaua de știri se mai cheamă "articole") pe oricare dintre servere: fluxurile din rețea asigură faptul că mesajul va ajunge, în cele din urmă, pe toate serverele. Pe de altă parte, **nu este obligatoriu** ca un articol să ajungă la toți potențialii săi cititori, deoarece "descărcarea" unui articol pentru citire se face din inițiativa utilizatorului. Clienții se conectează la serverul cel mai convenabil (sau la care au acces) și pot accesa orice newsgroup, pot citi doar articolele care-i interesează, pot să posteze articole pe oricare newsgroup sau chiar pe mai multe simultan (*cross-posting*). Forumurile rețelei de știri sunt, deci, **deschise**.

Eficiența acestei rețele de știri (numită Usenet) provine din faptul că serverele sale (de regulă mașini Unix foarte puternice) sunt conectate prin linii de comunicație de mare capacitate. Trebuie făcută o precizare: Use-

Jargon

Câteva expresii consacrate în forumurile din Rețea:
AFAIK (*as far as I know*) - după câte știu;
BTW (*by the way*) - apropo;
CUL (*see you later*) - ne vedem mai târziu;
FYI (*for your information*) - pentru informarea ta/voastră;
FYA (*for your amusement*) - pentru amuzamentul tău/vostru;
IIRC (*if I recall correctly*) - dacă bine-mi amintesc;
IMHO (*in my humble opinion*) - după umila mea părere;
IOW (*in other words*) - cu alte cuvinte;
OTOH (*on the other hand*) - pe de altă parte;
ROTFL (*rolling on the floor laughing*) - mă prăpădesc de râs;
WRT (*with regard to sau with respect to*) având în vedere;
WTF (*what the ...?*) - ... (exprimă o mirare interogativă);
WTH (*what the hell?*) - ce naiba?

net nu este parte integrantă în Internet. Există noduri Internet care nu sunt în Usenet și există noduri Usenet care nu sunt în Internet.

Ar mai fi de făcut o precizare: Usenet nu este administrat de o instituție sau de un for central, ci de comunitatea membrilor săi. Aceasta face ca unele dintre afirmațiile "ferme" de mai sus să trebuiască ușor nuanțate. De pildă, așa cum vom vedea în continuare, nu toate serverele Usenet transportă toate grupurile tematice și nu orice mesaj este distribuit global.

Newsgroups

Newsgroup — grupul tematic, grupul de știri — este denumirea unui forum din Usenet, construcția de bază a acestuia. Este greu de spus cu certitudine câte grupuri tematice există în Usenet. Se pare că numărul acestora se apropie la ora actuală de 50.000 și în fiecare zi apar grupuri noi. Pentru a sistematiza cât de cât acest imens număr de forumuri, s-a recurs la o organizare ierarhică, asemănătoare oarecum structurii sistemului de fișiere. Denumirea unui grup exprimă astfel poziția grupului în ierarhie.

Numele unui grup tematic începe cu un cuvânt ce marchează tematica generală a grupului (rădăcina), urmat de alte noi determinări, delimitate prin punct. Astfel, grupul denumit *comp.delphi* cuprinde discuții pe tema larg răspândităi platformei de programare Delphi de la Borland. Ierarhia poate fi mai mult sau mai puțin adâncă, în funcție de diversitatea tematicilor înrudite. Există, de exemplu, un newsgroup *comp.lang.pascal*, care găzduiește discuții generale despre limbajul de programare Pascal, dar există și grupul mai specializat numit *comp.lang.pascal.borland*, în care se discută doar despre variantele de Pascal promovate de firma Borland.

În timp, s-au "standardizat" 7 domenii tematice mari, adunate sub numele de "*UseNet hierarchy*" (ierarhia Usenet). Ele denumesc rădăcinile celor șapte ramuri "oficiale" ale ierarhiei (uneori le vom numi și pe ele ierarhii):

comp (*computers*). Teme legate de calculatoare. Ele au fost grupurile Usenet originale. Găzduiește discuții tehnice, la care iau parte în egală măsură savanți, specialiști, studenții sau simpli amatori.

news (*newsgroups*). Teme legate de însăși grupurile tematice. Aici se discută despre organizarea ierarhiilor, despre înființarea și desființarea unor grupuri, aici sunt sfătuiți novicii (*newbies*) să se "documenteze", etc.

sci (*science*). Este domeniul științelor, fiind populat mai ales de savanți și specialiști. Cu toate că este mai degrabă orientat spre științe exacte (există numeroase ramificații spre matematici, fizică, chimie etc.) și inginerești (agricultură, aeronautică, electronică etc.), aici sunt găzduite și grupuri dedicate arheologiei, antropologiei, filozofiei, psihologiei etc.

rec (*recreational*). Este ierarhia "timpului liber". Aici se găsesc grupurile tematice ale colecționarilor de tot felul, cele ale fanilor diverselor genuri muzicale sau ale unor muzicieni celebrii, sporturi și sportivi, grădinărit și animale de casă, hobby-uri de tot felul...

soc (*society*). Probleme sociale, istorie, religii, cultură în general și culturi naționale și regionale etc.

talk (dezbateri). Aici sunt găzduite forumuri care dezbat probleme controversate din domeniul politic (de pildă cele legate de criptografie, Uniunea Europeană,

Câteva sugestii în Usenet

comp.ai.philosophy - Aspecte filozofice ale inteligenței artificiale.

comp.arch - Despre arhitectura calculatoarelor.

comp.compilers - Compilatoare - structură, teorie (cu moderator).

comp.compression - Algoritmi de comprimare a datelor.

comp.databases - Despre administrarea bazelor de date și a datelor în general.

comp.graphics - Grafică, artă, animație.

comp.lang.c - Despre limbajul C.

comp.lang.c++ - Conversații pe tema limbajului C++.

comp.object - Despre programarea orientată pe obiecte în general.

comp.programming - Toate subiectele posibile legate de programare.

rec.humor - Glume și anecdote, unele un pic agresive.

rec.humor.funny - Față de newsgroup-ul anterior, acesta este moderat, deci calitatea este asigurată.

sci.astro - Astronomie.

sci.crypt - Metode de codificare/decodificare.

sci.math - Matematică de nivel mediu și superior.

sci.space - Despre spațiul cosmic, programele și cercetarea spațială.

talk.bizarre - Cele mai neobișnuite, ciudate, curioase sau stupide subiecte.

alt.books.isaac-asimov - Despre "regele" literaturii SF, regretatul Isaac Asimov.

alt.hackers - Ce isprăvi au mai făcut acești guru ai virtualului? (Automoderată)

(D.M.)

problema Tibetului etc.), social (eutanasia, avortul) și religios.

misc (*miscellaneous*). Aici se găsesc grupuri dintre cele mai diverse, care fie nu se pot încadra în celelalte ierarhii, fie s-au stabilit aici mai demult și... au rămas (sănătate, juridic, educație etc.).

La acestea se mai adaugă în unele clasificări și grupul *humanities* (dedicat științelor umane), formând astfel o clasificare numită "Big Eigth". Singura rațiune pentru care rădăcina *humanities* nu apare în ierarhia "oficială" este faptul că este încă destul de puțin populată, dovedind reținerile oamenilor de știință din domeniul științelor umane în privința utilizării mijloacelor moderne de comunicare.

Există o rădăcină mai specială, care se numește **alt** (*alternative*). Ea nu se supune nici unei reguli, găzduind de-a valma cele mai diverse grupuri tematice, de la cele mai serioase (de genul *alt.education.distance* sau *alt.politics.economics*), trecând printr-o întreagă pleiadă de hobby-uri (de la simplul *alt.beer* și până la *alt.prophecies.nostradamus* sau *alt.tv.muppets*), până la cele mai trăznite (*alt.conspiracy.usenet-cabal* sau *alt.happy.birthday.to.me*). Și multe, multe altele (nu toate la fel de inofensive)... De asemenea, fluxurile care transportă conținutul acestor grupuri este alternativ iar procesul de creare a grupurilor este diferit. Andrew Tanenbaum folosește o comparație sugestivă: "*Alt* este față de grupurile generale ceea ce este un talcioc față de un supermarket".

Mai există și alte arborescențe care nu se încadrează în ierarhia principală. Ele sunt cele așa-zis "minore" și cele locale.

Dintre cele "minore" unele sunt destul de importante și populate:

biz (*business*) — legate de afaceri (de pildă *biz.comp.services*);

bionet (*biology*) — dedicate biologilor;

vmsnet — discuții asupra sistemelor VAX/VMS;

gnu — legate de programele GNU dezvoltate de Free Software Foundation;

k12 (*kindergarten through 12th grade*) — grupuri tematice dedicate învățământului și educației pre-universitare.

În fine, ierarhiile locale sunt de interes regional (de pildă la nivelul unor state sau orașe din SUA) sau național (caz în care se desfășoară în limbile respective).

Lista se poate extinde la nesfârșit. În fiecare zi apar categorii și grupuri tematice noi, unele de interes regional, altele de interes larg.

Crearea unui nou newsgroup este guvernată de o procedură care prevede un anunț și o dezbatere în anumite grupuri din ierarhia *news*, urmate de o procedură de votare desfășurată prin e-mail (pentru a fi controlabilă). Crearea grupului este acceptată dacă cel puțin două treimi din votanți (subiecți neutri, neimplicați) sunt de acord și, în plus, numărul voturilor "pentru" depășește cu cel puțin 100 numărul voturilor "contra".

În fine, ar mai trebui spus că există și grupuri de discuție "moderate", în sensul că există un moderator

Neticheta în Usenet

Câteva reguli de comportare în Rețea, extrase dintr-un FAQ postat recent în grupul *news.announce.newusers*:

Nu uita niciodată că persoana de la celălalt capăt este un om.
Interacționând prin intermediul rețelei și a calculatoarelor, tinzi să uiți că de fapt discuți cu oameni. Nu lăsa supărarea să te stăpânească dacă nu poți convinge prin argumente. Jignirile nu rezolvă nimic și creează noi probleme.

Niciodată să nu pornești de la premiza că o persoană vorbește în numele organizației sale.
Multe persoane postează de pe calculatoarele de la serviciu, așa că în antet apare și numele organizației. Considerați mesajele ca fiind personale, dacă cel care le-a postat nu a precizat altceva.

Ai grijă ce spui despre ceilalți.
Nu uita că alături de tine mai citesc știrile câteva milioane de oameni. Printre ei s-ar putea să se numere și șeful tău sau șeful prietenului tău, cel mai bun prieten al fratelui iubitei tale sau prietenul de pahar al tatălui tău. Gândește-te de două ori înainte de a posta informații personale despre tine sau despre alții.

Fii scurt.
Nu spune zece vorbe dacă poți spune mai puține. Nu uita: cu cât e mai lung articolul, cu atât îl vor citi mai puțini.

Mesajele tale te reprezintă: fii mândru de ele.
Cei mai mulți oameni de pe Usenet nu se cunosc decât prin ceea ce au postat. Într-o zi ți-ar putea deveni colegi sau prieteni. Gândește-te bine dacă ceea ce postezi nu te-ar putea pune vreodată într-o situație jenantă. Deoarece scrisul te reprezintă, încearcă să scrii corect și inteligibil.

Folosește titluri descriptive.
Timpul e o resursă limitată, așa că dacă articolul tău nu are un titlu suficient de sugestiv, prea puțin îl vor citi.

care urmărește discuțiile și încearcă să prevină conflictele. Despre aceste liste se spune că au un "raport semnal/zgomot" mai bun decât cele ne-moderate (imensa majoritate).

Dialogul propriu-zis

Utilizarea rețelei de știri seamănă în foarte multe privințe cu utilizarea poștei electronice pentru a participa la listele de discuții. Există însă unele aspecte diferite, asupra cărora merită să insistăm puțin.

Conectarea. Spre deosebire de e-mail, unde toate serviciile de administrare sunt asigurate de furnizorul de servicii Internet (ISP — *Internet Service Provider*), știrile Usenet pot să fie sau să nu fie parte integrantă din serviciile oferite de ISP. Este posibil ca furnizorul de acces Internet să nu aducă deloc știrile Usenet pe serverele locale sau să aducă doar o parte din grupurile tematice. Dacă știrile sunt oferite de furnizorul de servicii Internet, atunci conectarea se face la adresa pe care acesta v-o furnizează. Dacă ISP-ul nu aduce știrile sau dacă grupurile care va interesează nu figurează printre cele pe care vi le oferă, există posibilitatea de a apela la o "sursă de știri" (*newsfeed*) gratuit. O listă completă poate fi găsită la http://www.freenewsgroups.com/. În fine, mai există câteva alternative care vor fi prezentate mai târziu.

Utilitare de acces. Clienții e-mail nu pot fi folosiți pentru accesul la grupurile de discuții din Usenet, deoarece protocolul de transfer nu este POP3, ci NNTP (*Network News Transfer Protocol*). Există numeroase programe de acces la știrile Usenet, numite în mod generic "cititoare de știri" (*news readers*). Ultimele versiune de navigatoare Web sunt dotate și cu cititoare de știri, dar există numeroase alte programe, dintre care unele gratuite sau shareware (vezi caseta dedicată programelor). În orice caz, cititorul de știri trebuie să ofere câteva seturi de funcții, dintre care cele mai importante sunt cele de **conectare și transfer** (categoria *Get — groups, headers, bodies*), cele de **navigare** (liniară și arborescentă — printre grupuri, anteturi, fire de discuție etc; de exemplu *View next unread message* sau *Skip to next unread message in thread*), cele de **postare** (pentru mesaje noi și replici — *follow-up*) și cele de **organizare** (incluzând aici funcțiile de marcare a mesajelor, de subscriere, de salvare etc.). Majoritatea cititoarelor de știri dispun și de funcții de **poștă electronică** (există posibilitatea de a expedia mesaje noi, de a răspunde prin e-mail la articole, de a "trimite mai departe" unele articole; de regulă nu și posibilitatea de a primi mesaje).

Abonamente. Deși termenul englezesc este același (*to subscribe*), în limba română tind să se impună doi termeni care-i corespund: "a (te) abona" (folosit mai ales pentru listele de discuții) și "a subscrie" (folosit mai ales pentru grupuri de știri). Disocierea este fericită, deoarece marchează o diferență semantică importantă: în timp ce abonarea la o listă de discuții implică o convenție cu serverul care găzduiește lista, subscrierea la un newsgroup este o informație locală:

Gândește-te la audiența mesajului.
Asigură-te că un mesaj pe care-l postezi într-un grup se încadrează în tematica acestuia. Dacă articolul este de interes local, limitează domeniul de distribuție. Mesajele de test postează-le doar pe *misc.test* sau *alt.test*, altfel riști să ți se umple mail-box-ul cu mesaje nu tocmai plăcute.

Fii atent cu umorul și sarcasmul.
În lipsa comunicării "față în față" și în condițiile unei audiențe diverse din perspectivă culturală, glumele s-ar putea să nu fie remarcate sau înțelese ca atare. Marchează-le cu *smiley*.

Postează un articol o singură dată.
Chiar dacă postezi pe mai multe grupuri, postează o singură dată (*cross-posting*). În felul acesta în rețea va exista o singură instanță a articolului, în toate grupurile. Mesajul odată citit într-un grup, nu va mai fi citit și în altele.

Rezumă problema la care răspunzi.
Când postezi un follow-up, rezumă pasajele din mesajul original la care răspunzi. În felul acesta, cititorii îți vor putea aprecia răspunsul în loc să se trudească să-și amintească ce spunea articolul original.

Citește tot firul de discuție înainte de a răspunde.
Nu răspunde la un articol înainte de a citi toate articolele din firul de discuție în cauză. Este posibil ca altcineva să fi dat deja răspunsul potrivit. Nu-l repeta.

Fii atent la copyright și licențe.
Drepturile de proprietate intelectuală sunt valabile și în rețea. Pentru o scurtă introducere poți să citești articolul "Copyright Myths FAQ: 10 big myths about copyright explained" la http://www.clari.net/brad/copymyths.html.

Nu folosi Usenet-ul ca să-ți faci temele.
Usenet-ul nu este o resursă potrivită pentru rezolvarea temelor de la școală. Dacă vei posta un chestionar sau o problemă "școlară", membrii grupului vor remarca imediat că ești un novice (*newbie*) și vei primi puține răspunsuri și multe ironii.

pur și simplu ne instruim cititorul de știri că suntem interesați de grupul respectiv (Usenet-ul nu ține evidența clienților săi).

Lectura. Odată ce aveți un *news reader* instalat și un *news feed* disponibil, nu vă mai trebuie decât un scaun confortabil și timp. La prima conectare va trebui să așteptați mai mult pentru că, în general, cititorul aduce toate grupurile disponibile și își formează o bază de date. După această primă conectare, aveți posibilitatea să accesați oricare dintre grupurile disponibile. Este indicat să subscrieți la grupurile care vă interesează și pe care vreți să le urmăriți, astfel încât comanda *Get New Headers in Subscribed Groups* (de regulă disponibilă în fiecare cititor de știri) să vă aducă mesajele noi. O altă operațiune de rutină este căutarea grupurilor nou apărute (*Get New Groups*), prin care baza de date se îmbogățește zilnic cu noi grupuri.

Spre deosebire de e-mail, atunci când accesați un newsgroup veți primi doar anteturile mesajelor disponibile. Pentru a aduce articolul propriu-zis va trebui să cereți explicit acest lucru, ceea ce în programele cu interfață grafică este foarte simplu (de regulă un dublu-clic pe antet, dar există și comenzi de navigare de genul *View Next Unread*). De obicei, mesajele citite sunt marcate ca atare. Unele programe dispun și de alte posibilități de marcare, de pildă pentru regăsire sau pentru păstrare. Marcajele pentru regăsire sunt importante în cazul programelor care lucrează *offline*, deoarece avem posibilitatea să specificăm astfel toate mesajele care vrem să fie aduse la proxima conectare. Marcajele pentru păstrare sunt importante

Comunitate și identitate

Primele "comunități virtuale" au fost – din rațiuni lesne de înțeles – cele ale informaticienilor. De altfel Usenet-ul a fost creat, la origini, în scopul schimbului de informații între utilizatorii de Unix (numele inițial era *Unix Users' Network*). Micile BBS-urile locale, găzduite de PC-uri conectate la rețeaua telefonică, au fost la început apanajul aproape exclusiv al *hacker*-ilor. Caracterul lor oarecum subversiv s-a pretat de minune la comunicarea de bug-uri de securitate, de chei secrete pentru "spargerea" diverselor programe comerciale, de adrese ale altor BBS-uri de același gen.

Dark Avanger

Pentru mulți dintre voi nume ca Dark Avanger sau Ping_Pong.B nu spun mare lucru. Pentru cei care lucrau pe PC-uri la începutul anilor '90, aceste nume provocau fiori: viruși bulgărești! Un context economic și politic special a făcut ca Bulgaria să devină la sfârșitul anilor '80 cel mai mare producător de viruși pentru calculatoare din lume. În acest fenomen, un rol central l-a jucat – ați ghicit deja – un BBS "clandestin", numit *Virus Exchange* și administrat de un "hacker anarhist" cunoscut sub numele de Commander Tosh. Mai târziu s-a vădit că numele său adevărat era Todor Todorov și că BBS-ul rula în apartamentul mamei sale, conectat printr-un singur modem de 2400 baud. BBS-ul a dobândit o tristă faimă în lumea întreagă, deoarece în jurul său s-a constituit un adevărat "club virtual". Ideea era că oricine putea să descarce de acolo coduri sursă pentru viruși, putea să le studieze, să învețe trucurile și să scrie alți viruși. Acesta era condiția pentru a accesa BBS-ul: trebuia să depună noi viruși. Poate aici s-au născut și virușii polimorfici, care se auto-transformă... Cert este că cel care-și zicea Dark Avanger nu s-a deconspirat niciodată.

The Well

În anul 1984, Larry Brilliant și Stewart Brand s-au întâlnit și au avut ideea de a constitui o grupare on-line, o comunitate virtuală care să poată să se întâlnească prin intermediul computerelor, să schimbe idei și opinii, să discute subiectele care-i preocupă. Gruparea s-a numit Well și a fost lansată la 1 aprilie 1985, într-o vreme când Usenet-ul era încă apanajul aproape exclusiv al informaticienilor. Constituit ca un fel de BBS mai mare, Well era accesibil la începuturi doar în California și doar celor care dispuneau de un modem.

Inima sa tehnică era un program numit PicoSpan, scris de un student al Universității din Michigan pe nume Marcus Watts. Deși a adus și multe noutăți tehnice, Well va intra în istorie mai degrabă prin comunitatea care s-a format aici și prin principiul pe care l-a consacrat: YOYOW (*You Own Your Own Words* – cuvintele tale îți aparțin): *"Ești proprietarul propriilor tale cuvinte. Acest fapt presupune că ești responsabil pentru cuvintele care le postezi în cadrul Well-ului și că reproducerea acestora fără permisiunea ta prealabilă în orice mediu exterior conferințelor din cadrul Well îți dă dreptul să-ți revendici drepturile tale de autor"*.

Fascinanta poveste a grupării Well o puteți citi în *Agora On Line* (http://www.aol.ro), în seria "Agora City" semnată de Emanuela Grama.

The Well există și astăzi. S-a mutat în Internet (http://www.well.com) și găzduiește 260 de conferințe. Ca și la începuturi, intrarea se plătește.

Stroustrup

Bjarne Stroustrup și-a asigurat deja un loc în istoria informaticii. Dacă totuși cineva n-a auzit de el: este cel care a conceput limbajul C++. La începutul acestui

întotdeauna, deoarece la închiderea sesiunii (sau, la unele cititoare, la părăsirea unui grup) articolele respective nu sunt păstrate local, decât dacă se specifică acest lucru sau se salvează explicit mesajele. Aceasta înseamnă că dacă vrem să revenim, tot transferul trebuie repetat.

O particularitate foarte importantă a grupurilor de discuție o reprezintă **firele de discuție** (*threads*). Un fir de discuție reunește toate mesajele care se referă la același subiect. Mai precis, reunește mesajele între care există referințe (așa cum vom vedea la postare). De obicei, acestea capătă o structură arborescentă. Practic, toate cititoarele moderne de știri, suportă fire de discuție. O particularitate interesantă este că articolele dintr-un *thread* sunt păstrate împreună indiferent de modul de sortare a articolelor (dacă cititorul permite sortarea).

Postarea. Așa cum am mai subliniat, oricine poate participa activ la discuțiile unui grup, indiferent dacă e abonat sau nu, publicând articole. Există două modalități de a posta un mesaj.

Prima cale este de a răspunde la un articol deja postat, ceea ce se cheamă ***follow-up*** (urmare, continuare). Este oarecum asemănător cu "răspunsul" (*reply*) prin e-mail. Mesajul va prelua automat câmpul Subject al mesajului la care răspundeți. Ceea ce este important este că mesajul pe care l-ați postat în acesta va fi încadrat într-un fir de discuție, împreună cu mesajul la care ați răspuns. Deoarece se poate posta un răspuns la un alt răspuns, structura unui *thread* poate să se desfășoare pe mai multe nivele. Este de menționat aici că relațiile dintre mesaje sunt sta-

an, Rețeaua i-a jucat o imensă farsă lui Bjarne: cineva a lansat în Usenet un text despre care pretindea că este finalul interviului acordat de Stroustrup revistei *Computer* a prestigioasei organizații IEEE (*Institute of Electrical and Electronics Engineers*). Se pretinde că textul este transcrierea brută, ne-prelucrată a conversației, care i-a uluit pe editori în așa măsură încât au refuzat să o publice. În cadrul discuției transcrise, marele Bjarne Stroustrup pretinde nici mai mult nici mai puțin că limbajul C++ a fost o glumă pe care lumea a luat-o în serios. Ideea de la care a plecat a fost că pe măsură ce numărul specialiștilor într-un limbaj crește, printr-un joc al cererii și ofertei, salariile plătite acestora tind să scadă. "Oare ce s-ar întâmpla dacă s-ar inventa un limbaj atât de complicat, atât de greu de învățat, încât nimeni să nu poată inunda piața cu programatori", s-ar fi întrebat Bjarne într-o zi... Și de aici vă puteți închipui singuri povestea.

Mesajul a fost reluat imediat de membrii grupului de știri unde a fost postat (*comp.lang.perl.misc*) și probabil în mai puțin de o zi s-a împrăștiat pe tot globul. Întreaga lume era uluită, dar nimeni nu putea să confirme sau să infirme autenticitatea textului. Scris cu mult har, dovedind o excelentă cunoaștere a lumii informatice și a mediului programatorilor, textul s-a dovedit a fi apocrif. Dar a pus o lume întreagă pe gânduri... Zvonurile circulă mai repede în Rețea.

McLuhan

Eseistul canadian Marshall McLuhan este autorul unor celebre studii despre efectele dezvoltării accelerate a mijloacelor de informare în masă în societatea modernă. Lucrările sale – dintre care cea mai cunoscută este *Galaxia Gutenberg* – au fost mereu vizionare și incitante, vehiculând concepte insolite ("satul global") și adesea (aparent) paradoxale ("mediul este mesajul"). Faima lui McLuhan nu a scăzut nici după moartea sa, în 1980.

Interesant este că dintre multele cărți pe care le-a semnat începând din 1951, doar cele dintâi sunt scrise efectiv de McLuhan. Asupra celorlalte plutește încă misterul și comunitatea academică este de acord că sunt, într-o măsură mai mică sau mai mare, compilații. McLuhan prefera să vorbească și uneori asistenții săi transcriau ce spunea după dictare, alteori transcriau din ce-și aminteau, alteori transcriau ce-au înțeles ei din ce spunea profesorul. Cărțile erau asamblate din notele sale și notițele asistenților. McLuhan semna cărțile fără să-și facă probleme, insistând adesea, în stilul său paradoxal, că faptele în sine nu sunt atât de importante ca falsele aparențe...

Prin 1994-1995, pe una dintre listele de discuții (*Zone*) găzduite de revista Wired, a început să posteze cineva care-și spunea Marshall McLuhan. Nimic deosebit până aici, pseudonimele sunt ceva obișnuit în comunitățile virtuale. Însă intervențiile sale exprimau într-un mod extrem de fidel modul de a gândi a lui McLuhan, imitându-i stilul până la perfecțiune. Intrigat, Gary Wolf – editorul executiv al revistei Wired – a consultat foști apropiați ai lui McLuhan, care au afirmat că dacă personajul nu era McLuhan în persoană, atunci nu putea fi decât duhul său. Cum McLuhan este considerat de Wired "patron spiritual" iar explozia Internetului n-a beneficiat de analizele lui, Gary Wolf a insistat până când a reușit să intre în corespondență cu așazisul McLuhan. Interviul – absolut fascinant – care a rezultat a fost publicat de Wired sub titlul *Channeling McLuhan* (http://www.wired.com/wired/4.01/features/channeling.html).

Falsul McLuhan pretinde că trăiește într-un orășel din sudul Californiei numit Parma. Nu există în zonă un oraș cu numele acesta. Probabil nu vom ști niciodată cine este în realitate. Dar în Rețea există doar prin ceea ce spui.

Mircea Sârbu

News Readers

Epoca în care se foloseau exclusiv cititoarele de ştiri cu linie de comandă a trecut definitiv. Sistemele Unix dispun în continuare de utilitare cu nume înspăimântătoare cum ar fi **rn**, **trn**, **nn** sau **tin**, dar interfeţele grafice reprezintă o comoditate la care nu mai putem renunţa. Din fericire, abundenţa de programe disponibile ne permite o largă libertate de alegere. Cititoarele pe care vi le prezentăm în aceste pagini sunt toate pentru Windows (pe 16 sau 32 de biţi), dar există şi cititoare Macintosh. O altă trăsătură comună tuturor programelor din grupaj este că sunt **gratuite** şi pot fi descărcate din Internet. Nu am inclus în această prezentare două produse care însoţesc de regulă browserele Web: Netscape Collabra şi Microsoft Outlook Express. Ele sunt suficient de răspândite ca să nu aibă nevoie de o reclamă în plus.

FreeAgent 1.11

Este varianta redusă a programului Agent, dezvoltat de Forte, Inc. Ca şi acesta, dispune de o interfaţă complexă, care poate să intimideze un utilizator începător, însă tocmai aici stă puterea programului.

Plusuri: Interfaţa, constând din trei panouri care nu se suprapun, este foarte ergonomică. Posibilitatea de a lucra off-line va fi foarte apreciată de utilizatorii care se conectează la Internet prin linii telefonice comutate. Marcajele diverse şi numeroasele funcţii de navigare sunt deosebit de utile.

Minusuri: Nu dispune de foldere. Nu poate sorta articolele. Posibilităţile de căutare a grupurilor de ştiri sunt limitate. Nu are funcţie *kill-file*.

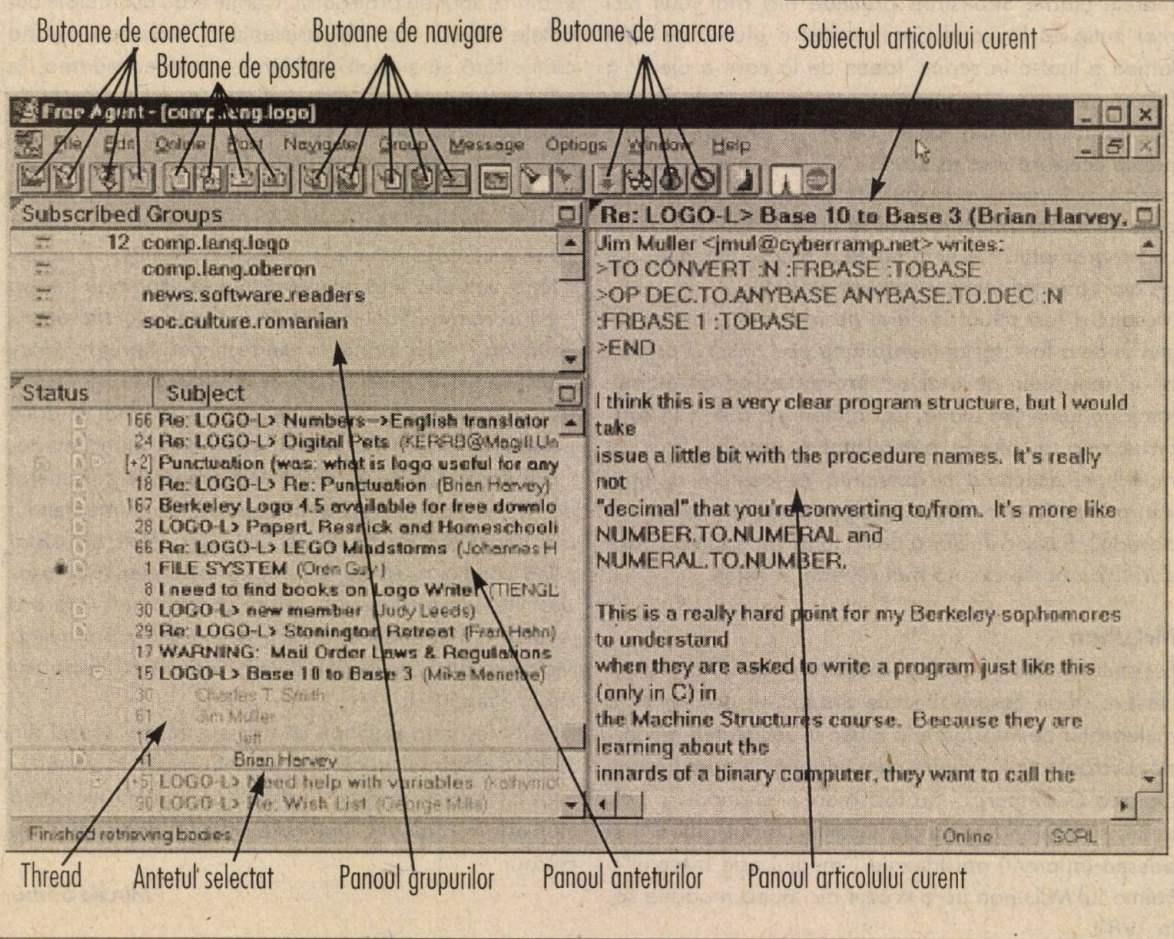
Download: http://www.forteinc.com/

bilite prin referinţe între identificatorii mesajelor — deci ele există indiferent dacă cititorul vostru de ştiri poate sau nu poate să reprezinte (de obicei prin "indentare") fire de discuţie cu structuri mai complexe.

A doua variantă este de a posta un articol "nou", adică de a deschide un nou subiect de discuţie. Dacă cineva răspunde, atunci acest articol va fi rădăcina unui nou fir de discuţie.

Există şi posibilitatea de a posta acelaşi articol în mai multe grupuri simultan (*cross posting*). De pildă, o noutate legată de o "scamă" pe care aţi descoperit-o într-o implementare Java poate fi trimisă la toate grupurile de discuţii legate de Java (sunt mai mult de 10). În această situaţie este important să specificaţi unde să fie trimise răspunsurile (câmpul *follow-up to*). Una dintre opţiunile de postare importante este distribuţia, care poate fi globală (*world*) sau limitată.

O menţiune specială legată de folosirea poştei electronice în lumea



grupurilor Usenet: așa cum am amintit deja, multe dintre cititoarele de știri oferă facilități în acest sens, astfel încât putem răspunde direct autorului pe această cale și putem trimite (*forward*) unor alte persoane anumite articole care credem că-i interesează. Importante sunt în acest context facilitățile complementare: atunci când postăm un *follow-up* la un articol avem posibilitatea să-i trimitem autorului articolului mesajul și prin e-mail (chiar în comanda de postare). De asemenea, putem să cerem ca toate răspunsurile la un articol pe care l-am postat să ne fie trimise automat prin e-mail (în câmpul *Follow-up to* vom pune cuvântul *posting*).

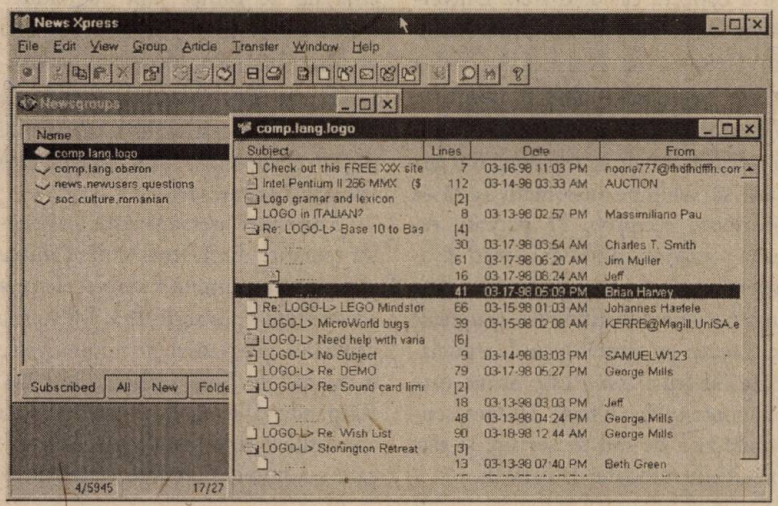
Neticheta. În cea mai mare parte, regulile elementare de netichetă sunt aceleași ca și pentru poșta electronică, prezentate în numărul trecut. Urmărirea discuțiilor dintr-un grup o perioadă de timp, înainte de a posta, este o idee bună. De asemenea, este recomandabil să citiți un document numit lista FAQ (*Frequently Asked Question*) a grupului, pentru a nu ridica din nou probleme de care membrii mai vechi s-au plictisit. Această listă este întreținută de un grup de voluntari și este postată la intervale regulate (de regulă între două săptămâni și o lună, în funcție de listă). Aceste FAQ-uri se publică și în câteva grupuri speciale: *comp. answers*, *news.answers* și *alt.answers*. Există, de asemenea, colecții de FAQ-uri disponibile în World Wide Web (una foarte bună o găsiți la http://www.faqs.org/). Există câteva FAQ-uri foarte bune chiar despre netichetă, pe care l-am folosit pentru caseta "Neticheta".

Internetul și Rețeaua

Distincția dintre Internet și Usenet poate să pară ciudată: multă lume crede că Internetul însumează toate rețelele interconectate din lume. Nu este adevărat: Internetul este doar una dintre inter-rețele din lume (e adevărat, cea mai mare). Pentru a desemna totalitatea rețelelor interconectate se folosește un termen mai larg: Rețeaua (*the Net*). Este pe de altă parte adevărat că Internetul tinde să

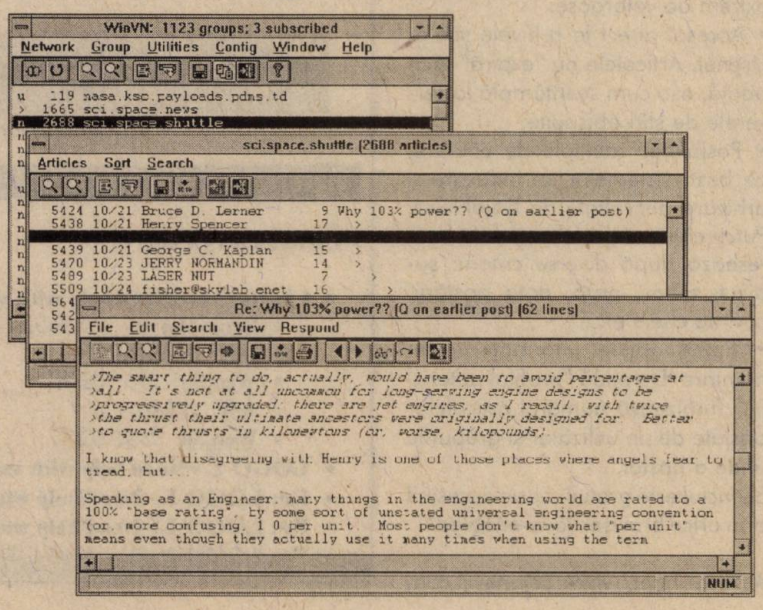
News Xpress 2.0

Este un program dezvoltat de către Ng Wang Lui, inițial ca un simplu exercițiu. Distribuit apoi în Internet ca freeware, a căpătat o popularitate considerabilă. Este foarte simplu de folosit, fiind recomandat începătorilor.
Plusuri: Dispune de un sistem de foldere pentru salvarea și organizarea mesajelor. Decodifică automat atașamentele binare. Suportă MIME și afișează imaginile JPEG. Suportă *kill-file*. Documentație excelentă.
Minusuri: Nu lucrează off-line. Fereastra grupurilor nu este ergonomică.
Download: ftp://ftp.malch.com/nx201.zip (și numeroase alte adrese...)



WinVN 0.99.9

Program dezvoltat inițial la NASA (Centrul spațial Kennedy) de către Mark Riordan. A devenit apoi *public domain*, dezvoltarea sa fiind în continuare coordonată de NASA. Sursele sunt disponibile.
Plusuri: Este rapid. Suportă MIME. Dispune de posibilități multiple de personalizare. Lucrează cu mai multe grupuri de discuții simultan.
Minusuri: Interfața cu ferestre multiple este incomodă.
Download: http://www.ksc.nasa.gov/software/winvn/



omogenizeze Rețeaua, cu atât mai mult cu cât actualmente aproape toate inter-rețele sunt, într-un fel sau altul, conectate la Internet (prin sisteme numite *gateways*).

Pentru subiectul prezentat în acest articol este, poate, relevant faptul că rețeaua Usenet a luat ființă în 1979-1980 și a fost implementată (de către Tom Truscott și Jim Ellis) pentru a conecta calculatoarele Universității Duke cu cele ale Universității din Carolina de Nord (UNC). S-a folosit protocolul UUCP (*Unix to Unix CoPy*) și a permis specialiștilor în calculatoare de la cele două universități să schimbe informații și să se informeze reciproc. Pe parcurs rețeaua s-a lărgit, mai întâi în mediile academice, dar apoi și în diverse instituții, atracția constituind-o grupurile de discuții pe teme legate de calculatoare (ierarhia *comp* este prima care s-a constituit). Astăzi Usenet, cuprinde zeci de mii de servere și este folosită de milioane de oameni.

O altă rețea academică internațională, Bitnet (*Because It's Time NETwork*), a avut un rol major în crearea comunităților virtuale, prin cele câteva mii de liste de discuții pe care le găzduiește (în 1989 Bitnet a fuzionat cu CSnet, constituind rețeaua CREN — *Corporation for Research and Education Network*). Actualmente, o parte dintre listele de discuții de pe Bitnet sunt "oglindite" (sau transferate) în Usenet, formând ierarhia *bit*. Nici Bitnet nu se confundă cu Internetul, deși nodurile rețelei sunt accesibile din Internet și poșta electronică din Bitnet este conectată la Internet.

Forumuri asemănătoare grupurilor tematice din Usenet sunt găzduite și de o rețea numită FidoNet. Istoria acestei rețele pornește de la BBS-urile utilizând programul de administrare dezvoltat de Tom Jennings în 1984. Răspândirea acestora a sugerat ideea constituirii unei rețele de BBS-uri, care s-a numit FidoNet. Actualmente, rețeaua reunește circa 30.000 de noduri din întreaga lume (conectarea se face prin linii telefonice). Forumurile din FidoNet se numesc "conferințe" și se bazează pe serviciul numit "echomail". Există astăzi "punți" între poșta electronică din Internet și cea din FidoNet și, mai de curând, posibilitatea de a accesa conferințele (foarte interesante!) din FidoNet prin WWW (o posibilitate: http://docsplace.dyn.ml.org/).

Odată cu creșterea popularității World Wide Web, au apărut tot mai multe forumuri pe Web. În principiu, acestea nu diferă de grupurile tematice din Usenet. Diferența majoră este că un forum Web este găzduit de un sit și poate fi accesat cu un browser. Toate operațiile obișnuite (navigarea printre articole, lectura, postarea) se fac cu browserul. O colecție impresionantă (peste 180.000) de forumuri Web puteți găsi la http://www. ForumOne. com. Există mai multe situri care oferă gratuit posibilitatea de a vă crea

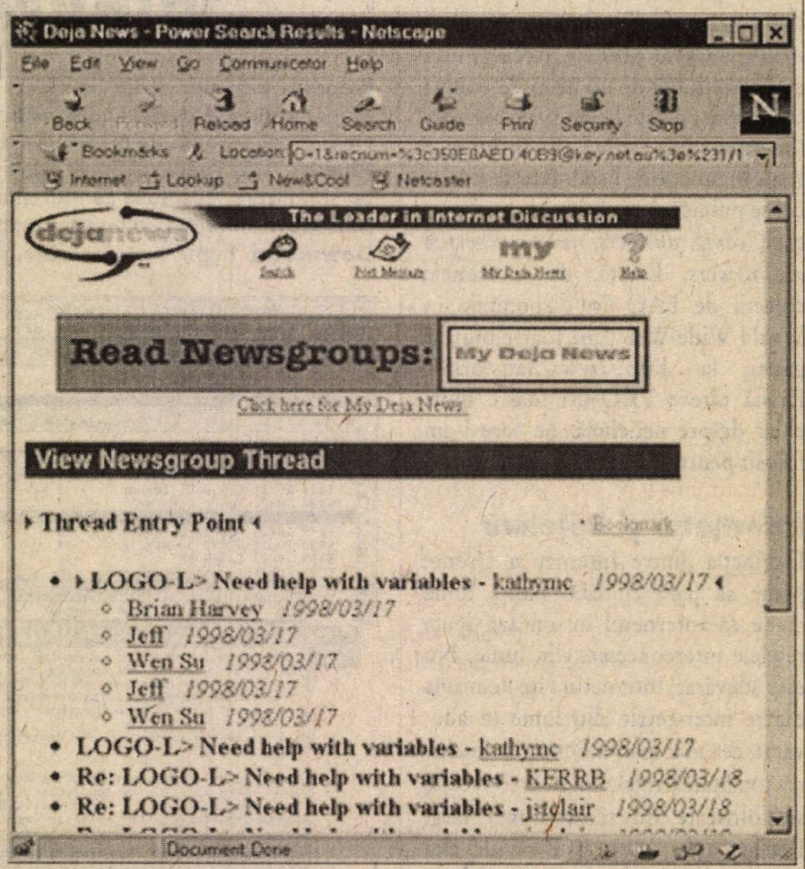
DejaNews

Un sit Web care oferă acces la Usenet printr-un *gateway* HTTP. Are dezavantajul că accesul este mai lent decât în cazul cititoarelor de știri obișnuite, în schimb oferă o serie de facilități suplimentare extrem de valoroase:

- Accesul direct la arhivele știrilor Usenet. Articolele nu "expiră" niciodată, așa cum se întâmplă la serverele de știri obișnuite.
- Posibilități multiple de căutare, pe baza sistemelor de indexare și arhivare dezvoltate de DejaNews. Puteți căuta articolele care vă interesează după diverse criterii: subiect, autor, grup, data postării, cuvinte cheie etc.
- Puteți obține informații suplimentare. De pildă "profilul autorului" furnizează numărul articolelor postate de un utilizator și grupurile unde a postat.

Serviciul este gratuit și este accesibil prin orice browser Java-enabled.

Adresa: http://www.dejanews.com



propriul forum. Cu toate acestea, forumurile de pe Web sunt încă departe de amploarea grupurilor de știri.

În schimb, Web-ul a adus știrile mai aproape de mulți dintre utilizatorii Internetului, prin posibilitățile de acces pe care le oferă la multe dintre arhivele grupurilor și chiar la întregul Usenet. Compania Deja News oferă prin situl său Web (http://www.dejanews.com) acces liber și gratuit la întregul Usenet. Toate funcțiile unui *news reader* sunt implementate prin applet-uri și, în plus, sunt disponibile arhivele grupurilor (un *news server* obișnuit nu păstrează știrile decât câteva zile), există diverse posibilități de căutare după diverse criterii, există servicii suplimentare (de pildă "profilul autorului", semne de carte etc.).

Comunități virtuale

În fiecare zi se postează pe Usenet articole care însumează aproape un gigabyte, ceea ce — ca text — înseamnă aproape cât *Encyclopaedia Britannica*. Fiecare articol este expus unui public potențial de câteva milioane de oameni. Dacă mai socotim și sutele de mii de alte forumuri din Rețea (o uriașă colecție de liste de discuție *doar* despre muzică puteți găsi la http://server.berkeley.edu/~ayukawa/lomml.html) avem imaginea unei planete a dialogului. Oameni care nu s-au întâlnit și nici nu se vor întâlni niciodată în viața reală, de pe continente diferite, aparținând unor culturi diferite, se întâlnesc în aceste forumuri și discută despre compilatoare sau despre pisici birmaneze, despre proiectarea roboților sau despre rețete culinare...

Se naște o solidaritate emoționantă. O învățătoare din Australia are o neclaritate despre variabilele locale din Logo și primește cinci răspunsuri pline de explicații și exemple, dintre care unul dat de însuși Brian Harvey de la Berkeley, un adevărat guru al domeniului... Sistemul de operare Linux a fost dezvoltat de câteva sute (sau poate mii) de programatori voluntari, care au dialogat prin grupuri de știri. Iubitorii muzicii lui Frank Zappa au locurile lor de întâlnire iar colecționarii de timbre le au pe ale lor. Cișmigiul șahiștilor amatori este transpus la scară planetară. Satul global prefigurat de Marshall McLuhan nu mai este o metaforă.

Pe de altă parte, este adevărat, există în Rețea agresivitate, rasism, pornografie, vulgaritate, impostură și minciună... Există *flame wars* și *spamming*. Așa cum există și în viața reală. La fel de adevărat este că nici un fel de cenzură nu poate afecta forumurile din Rețea. Libertatea este îngrădită doar de responsabilitate.

Rețeaua are multe fațete și discernământul este obligatoriu în momentul în care pășiți în lumea aceasta uluitoare și fascinantă. Virtuală și reală în același timp.

❑

Mircea Sârbu este directorul editorial al editurii Computer Press Agora. Poate fi contactat prin e-mail la adresa: msarbu@agora.ro.
Dragoș Marian este colaborator al GInfo. Îi puteți scrie la redacție: office@ginfo.ro

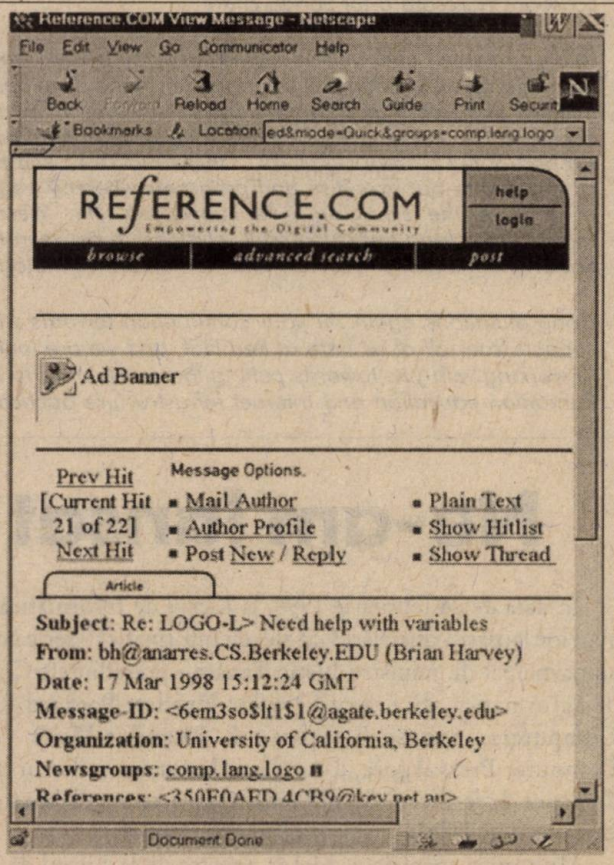
Reference

Reference.COM a luat naștere pe baza unui parteneriat între Unversitatea Stanford și InReference, Inc. Serviciile oferite de Reference sunt în bună măsură asemănatoare cu cele de la DejaNews în ceea ce privește știrile Usenet, fără a avea însă interfața atât de cizelată. În schimb, Reference nu se mărginește la Usenet, ci se extinde asupra tuturor formelor de comunicație în grup din Internet. Astfel, alături de grupurile de știri, Reference dispune de indexuri ample pentru liste de discuții și forumuri Web. În total, Reference arhivează peste 150.000 de forumuri (newsgoups, mailing lists și Web forums), fiind poate cea mai completă referință actuală în domeniu.

Reference se distinge și prin facitițăile deosebite de căutare de care dispune. Pe lîngă căutările simple, asemănătoare celor din Web, există și căutări mai sofisticate, după diverse criterii. O facilitate foarte utilă o constituie șabloanele predefinite de căutare pentru un mare număr de subiecte.

Cea mai atractivă facilitate oferită de Reference este însă "căutarea activă". Această facilitate permite utilizatorilor înregistrați (serviciile sunt gratuite) să monitorizeze pasiv anumite subiecte sau forumuri, rezultatele fiind livrate automat prin e-mail.

Adresa: http://www.reference.com



Linux pentru școli

Am aflat cu bucurie de la domnul Nicolai Sandu (coordonator programe Internet la Fundația pentru o Scietate Deschisă) că firma Red Hat Software va face Fundației o donație constând din 1800 de pachete de Red Hat LINUX 5.0, în valoare de peste 100.000 US$.

Dorința exprimată de donator este ca pachetele să fie distribuite de către Fundație, în cadrul programelor pe care le desfășoară, cu precădere în licee. Pachetele vor fi donate școlilor care doresc să-l instaleze, în special pentru comunicare (e-mail, Internet).

Fundația ne-a mai informat că pregătește un buget pentru ca, odată cu distribuirea pachetelor, să se facă și cursuri de instruire cu administratorii de rețele din licee. Aceste cursuri vor fi organizate la liceele interesate, de preferință cele care au deja o experiență în domeniu și care funcționează deja ca noduri Internet și pentru alte școli. Și-au exprimat deja disponibilitatea de a organiza astfel de cursuri licee din Focșani, Alba Iulia, Eforie Sud, Timișoara, Cluj, Iași, Oradea, Călărași, București.

Editura Computer Press Agora și-a exprimat disponibilitatea de a sprijini această acțiune prin intermediul revistei noastre. GInfo va mediatiza evenimentul și vă va ține la curent cu desfășurarea acțiunii.

Un rol important în obținerea acestei importante donații l-a avut domnul Cristian Gafton, fost administrator de rețea la Fundația Soros din Iași - actualmente angajat al companiei Red Hat - care a adus un important aport la promovarea mișcării Linux în România.

Nu în ultimul rând trebuie menționat numele domnului Robert Young, președintele companiei Red Hat Software, din a cărui scrisoare adresată d-lui Sandu citām:

"With a talented and well-educated work force there is no reason that Romania cannot become a world leader in the development of Internet and Information technologies. However, to do this you need access to the source code of these leading technologies, and the support of the current suppliers in these markets. Red Hat Software can help with our expertise and programs whereby we give you real control over the technologies we supply you with. We are looking forward to a long and successful relationship between the Open Society Foundation, Red Hat, and Romanian technology users.

Congratulations again on your contribution towards winning the support from all of us here at Red Hat, and we are looking forward to working with you towards putting these resources to effective use in Romanian education and Internet infrastructure applications."

Ne-am lansat

În data de 26 februarie 1998, la Liceul de Informatică din București, a avut loc lansarea oficială a Gazetei de Informatică - serie nouă. La festivitate au participat dl. ministru **Iustin Tanase** - președintele Comisiei Naționale de Informatică, directorii firmelor **Hewlett-Packard** Romania și **Forte Computers** - sponsorii revistei, dl. **Romulus Maier** - director general Computer Press Agora, d-na **Clara Ionescu** - redactor-șef, elevi și cadre didactice de la Liceul de Informatică, prieteni, colaboratori, Televiziunea Națională și Tele7abc. Le mulțumim tuturor.

Mulțumiri speciale gazdelor noastre de la Liceul de Informatică.

(*Continuare din pag. 64*)

Magdalena Stoica

Sunt Magdalena Stoica, profesor la liceul teoretic "Constantin Brâncoveanu" din orașul Titu, județul Dâmbovița. [...]Am primit primul număr al revistei GInfo, l-am citit — eu și elevii interesați. Elevii au sesizat primii nivelul deosebit de ridicat al problemelor propuse. Ca profesor, mi se părea normal ca o astfel de revistă să conțină lucruri deosebite. Ei ar fi vrut însă și probleme mai simple, așa cum se găsesc în culegeri de probleme pentru alte domenii. [...].

(1) Poate ar trebui rubrici cu probleme pentru admitere, alături de rubricile cu probleme pentru olimpiade.

(2) Foarte interesante, pentru noi, au fost articolele legate de Linux și de e-mail. Avem destul de puțină bibliografie pentru ele. Rugămintea ar fi ca, [...] să mai publicați astfel de articole.

(1) Problema accesibilității revistei în general și a problemelor propuse în special este una destul de delicată. Rezolvarea ei ține în mare măsură de reacțiile pe care le primim. Oricum, se pare că foarte curând vor apărea pe piață și culegeri de probleme de programare. Nu ne propunem să le înlocuim.

(2) Internetul va fi mereu în atenția redacției. Este o resursă imensă și accesul școlilor românești la Internet este o problemă vitală. Pentru că bibliografia există în Rețea, dar multe școli nu au acces la ea. Cât despre Linux... să nu anticipăm.

Dialogul continuă

Lista de discuții "dialog" funcționează. Micile probleme tehnice au fost depășite, așa că vă puteți înscrie trimițând la adresa majordomo@ginfo.ro un mesaj conținând doar linia "*subscribe dialog*".

Așteptăm în continuare scrisorile voastre pe adresele din caseta redacției.

Mircea Sârbu

Vă place GInfo ?

Nume: Prenume: Vârsta:

Adresa:

Localitate: Cod poștal: Județ: Telefon:

Ocupația: elev informatică ☐ elev ☐ student ☐ cadru didactic ☐ altă ocupație ☐ (precizați:)

Școala (firma, organizația etc.):

Dotări

Folosiți un calculator acasă? Da☐ Nu☐

Câte calculatoare sunt instalate în școală: (aprox.)

Aveți acces la e-mail: Da ☐ Nu☐

Aveți acces la WWW: Da ☐ Nu☐

Cum apreciați dotarea cu tehnică de calcul a școlii:

Aprecieri generale

Acordați note de la 1 la 10 pe următoarele criterii:

Structura tematică:

Calitatea articolelor:

Prezentare grafică:

Calitate tipografică:

Considerați că prețul este:

prea mare ☐ accesibil ☐ prea mic ☐

De cât timp cunoașteți Gazeta de Informatică (ani)?

Cum ați procurat acest număr:

chioșc ☐ abonament ☐ cadou ☐ altfel ☐

În acest număr

Apreciați cu note de la 1 la 10 articolele acestui număr:

Algoritmul Boyer-Moore

Durata de execuție a algoritmilor

Fotbal, software și roboți

JAVA - a doua lecție

Problema lui Steinhaus și Sierpinski

Cum s-a născut JAVA

Monica de la Microsoft

News: informație și opinie

Articolul care v-a plăcut cel mai mult:

De ce?

Articolul care v-a plăcut cel mai puțin:

De ce?

Considerați că ponderea problemelor rezolvate este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea problemelor propuse este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea subiectelor teoretice este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că nivelul de accesibilitate al articolelor este

prea cobotât ☐ potrivit ☐ prea ridicat ☐

În viitor

În următoarele numere așteptați:

mai multe articole de programare? Da ☐ Nu ☐

mai multe articole teoretice? Da ☐ Nu ☐

mai multe articole despre Internet? Da ☐ Nu ☐

mai multe prezentări de programe? Da ☐ Nu ☐

prezentări de jocuri? Da ☐ Nu ☐

articole despre hardware? Da ☐ Nu ☐

V-ar plăcea să citiți:

pagini redactate de elevii unor licee? Da ☐ Nu ☐

lucrări premiate, realizate de elevi? Da ☐ Nu ☐

actualități școlare? Da ☐ Nu ☐

prezentarea unor situri Web? Da ☐ Nu ☐

tips & tricks? Da ☐ Nu ☐

Comentarii:

Dacă nu vă este suficient spațiul, folosiți coli albe suplimentare.

Așteptăm scrisorile voastre!

Poșta redacției

Așa cum v-am promis, am așteptat reacțiile voastre. Și n-am așteptat degeaba. Am primit o mulțime de mesaje, așa că a trebuit să recurgem la o selecție. Numerotarea paragrafelor am adăugat-o pentru a putea răspunde "la obiect".

Sebastian Ghiurca (Galați):

(1) În urmă cu ceva timp am primit numărul unu din revista dvs. Prima impresie nu a fost foarte bună. Într-un număr mic de pagini ați încercat să acoperiți un număr mare de domenii și nu a ieșit prea bine. Deci, ideea este că se poate și mai mult.

(2) Am rămas profund indignat de articolul pe care l-a scris domnul Mihai Budiu, pe care îl admir pentru articolele publicate în PC Report. Mai ales de remarca din cuprins: "chiar știați totul despre tipuri de date?".

(3) Stimate domn, ceea ce scria în acel articol se învață la orice liceu în clasa a noua!!! (într-adevăr, despre pointeri se învață de-abia într-a zecea). Articolul a fost mult prea pueril și pentru o revistă de informatică scoasă la Văscăuții din Deal. Îmi pare rău pentru asemenea cuvinte, dar acesta este adevărul. Sper să țineți seama de acestea.

(1) Ne vom da silința. Singura observație este că nu încercăm să "acoperim" toate aceste domenii, ci doar să le abordăm, să atragem atenția asupra lor și să indicăm câteva surse pentru informații suplimentare. Creșterea numărului de pagini va veni cu timpul. Sperăm că "împărțirea paginilor" să mulțumească o parte cât mai mare dintre cititori.

(2) Remarca din cuprins nu îi aparține lui Mihai Budiu. În general, redacția intervine în "elementele adiționale" ale textului propriu-zis al unui articol (este o practică comună în redacții). Titlurile, subtitlurile, "pălăria" etc. sunt adesea "retușate". Ele au rolul de a atrage la lectură (chiar cu riscul de a intriga...), de a ajuta la accentuarea unor elemente, de a îmbogăți aspectul vizual al paginii ș.a.m.d.

(3) Există și elemente interesante în articolul incriminat. Noțiunile de polimorfism și supraîncărcare facilitează introducerea unor elemente de programare obiectuală... În plus, revista nu este destinată **exclusiv** elevilor din clase de informatică.

Oricum, vom ține seama de opinia dvs.

Premii

Prima etapă a concursului rezolvitorilor de probleme a fost câștigată, la egalitate, de Florin Sasu de la Liceul Spiru Haret din București și Daniel Popa din Orăștie, județul Hunedoara.

Pentru "cea mai bună idee" l-am ales luna aceasta pe Adrian Zaharia (*Ady din Severin*) care ne-a dat o idee interesantă: de a pune în contact cititorii cu interese comune în domeniul informaticii. O vom pune curând în aplicare.

Toți trei vor primi din partea noastră câte un CD-ROM. Pentru că nu știm care dintre cele disponibile îi interesează, îi rugăm să ia legătura cu redacția prin e-mail (sau eventual telefonic: 065-166.516).

Și voi puteți câștiga!

Daniela Conțiu

Bună, sunt Dana și îmi place informatica! Dacă vreți să îmi scrieți, o puteți face pe adresa: danico@lbvi.sfos.ro. Abia astept!

PS: Îmi mai place, de asemenea, și MariMar!

Hello Dana, toți cei care citesc GInfo iubesc informatica, așa că sunt sigur că vei primi multe mesaje.

Adrian Zaharia

Sunt Ady din Severin, elev la liceul Decebal, secția de info.

Tocmai am vizitat pagina ta de web, e fainâ, dar lipsește ceva: la probleme lipsește enunțul... Eu nu prea pot să fac rost de revistă, știi cum e si cu distribuția, așa că o iau direct de pe web. Deci, dacă se poate ca în viitor să apară și textul problemei rezolvate, ar fi perfect...

Din păcate, vei putea să iei revista de pe Web doar cu o întârziere de 2-3 luni. Explicația este simplă: pentru a realiza revista asta se cheltuiesc o mulțime de bani (e vorba de hârtie, manopera tipografica, comisionul distribuitorilor, regia, impozite si taxe, transport, drepturile de autor ale colaboratorilor, și multe, multe altele). Banii aceștia trebuie câștigați din două surse: 1. vânzarea revistei; 2. vânzare de spațiu publicitar in revista. Concluzia o poți trage singur: dacă revista nu se vinde...

Publicarea surselor pe Web imediat după apariției revistei o facem în scopul de a te scuti de dactilografierea codului, în cazul în care vrei să încerci programul (sau să-l modifici, să-i testezi performanțele în comparație cu rezolvarea ta etc.).

În schimb, intenționăm să punem pe Web tot felul de alte lucruri, care nu apar în revistă. Sper ca în curând pagina Web să devină la fel de atractivă ca și revista (pe care nu o va reproduce, ci o va completa).

(*Continuare în pag. 62*)

Ești pregătit pentru mâine?

COMPUTER PRESS AGORA

Abonează-te și vei fi!

Completați talonul de mai jos sau cartea poștală din revistă

Numele (Firma): _____

Prenumele: _____

Adresa: _____

Cod poștal/Localitate: _____

Tel./Fax: _____

Am avut abonament în 1997 Da ❑ Nu ❑

V-am expediat _____ lei, reprezentând contravaloarea abonamentului la revistele _____

_____ pe o perioadă de _____ luni, cu mandatul poștal/ordinul de plată nr. _____

din data _____, pentru C.P. Agora, în contul: 2021400027100178, Banca Transilvania, filiala Mureș.

❑ Doresc să-mi trimiteți factură

Data _____ Semnătura _____

		Preț vânzare	Preț abonament	Abonament elevi/studenți*	Luna din care doriți să înceapă abonamentul
BYTE România	12 luni	120.000 lei	96.000 lei ❑	72.000 lei ❑	
	6 luni	60.000 lei	48.000 lei ❑	36.000 lei ❑	
PC Report	12 luni	96.000 lei	76.800 lei ❑	57.600 lei ❑	
	6 luni	48.000 lei	38.400 lei ❑	28.800 lei ❑	
CAD Report	6 apariții	48.000 lei	38.400 lei ❑	28.800 lei ❑	
	3 apariții	24.000 lei	19.200 lei ❑	14.400 lei ❑	
Ginfo	5 numere**	25.000 lei	25.000 lei ❑	20.000 lei ❑	

*Elevii și studenții vor trebui să atașeze comenzii o copie după carnetul de elev sau de student

** În anul școlar 1998

HEWLETT® PACKARD

Seria de imprimante HP DeskJet 690 cu PhotoREt. Imaginea afacerilor rentabile.

Legenda spune că o imprimantă nu poate însufleți imaginile. Datorită unei strălucite combinații de tehnologii avansate, seria de imprimante HP DeskJet 690 spulberă odată pentru totdeauna această prejudecată. Imaginile imprimate au calitate fotografică datorită tehnologiei HP PhotoREt. Sistemul RealLife imaging asigură, cele mai vii culori şi cea mai bună calitate pentru reproducerile alb-negru. Mai mult decât atât! ColorSmart reglează automat nivelurile de culoare. Cu cartuşul Photo Cartrige, se obțin cele mai naturale culori, iar imaginile tale beneficiază de o calitate fotografică excepțională şi de o mai mare rezoluție. Nici n-apucă bine balaurul să pocnească din coadă şi documentele tale sunt deja tipărite, 5 pagini alb/negru sau 2 pagini color pe minut! Imprimanta HP DeskJet 690C reproduce imagini atât de reale de parcă simți arsura flăcărilor scoase de balaur. Aşa că, uită poveştile! Seria DeskJet 690 nu este una oarecare. Este sufletul documentelor tale!

IMPRIMANTELE HP. HÂRTIA ESTE ÎN SLUJBA TA.

DEALERI – Bucureşti: AMERILEX - 223 43 23 • MBL COMPUTERLAND - 336 75 99 • Q'NET - 211 78 55 • S&T ROMANIA - 330 73 20 • SYSTEM PLUS - 210 85 70 THEMA - 231 29 55 • **Bistriţa:** LOGITEC COMPUTER - 063/23 27 96 • **Cluj:** SISTEC-CLUJ - 064/19 02 82 • **Timişoara:** BRIDGEMEN - 056/20 13 60
RESELLERI – Bucureşti: ALSYS DATA - 411 27 27 • BLUE RIDGE INTERNATIONAL - 210 68 28 • CIPRO - 220 04 00 • FLAMINGO COMPUTERS COM - 222 49 34 • FOCUS INTERNATIONAL - 322 30 69 NET CONSULTING - 638 33 45 • PRODUCTON - 312 30 78 • RADIX - 410 25 25 • SOFTWARE ITC - 232 44 71 • TRICORP ELECTRONICS - 679 48 79 • TOTAL SOFT - 230 09 76
Baia Mare: CUMULUS LOGIC - 062/43 07 35 • **Bacău:** MIDO IMPEX - 034/17 10 80 • **Braşov:** ICCO COMPUTERS - 068/14 43 54 • INTELPROF INTERNATIONAL - 068/41 13 54 SISTEM - 068/31 86 63 • **Cluj Napoca:** AT COMPUTER - 064/41 32 04 • **Craiova:** AS COMPUTERS - 051/16 32 80 • **Constanţa:** FORTE DIGISOFT - 041/65 30 36 • GMB COMPUTERS - 041/61 92 22
Deva: ASCOPIER - 054/23 36 06 • **Galaţi:** ROMLOTUS - 036/47 40 00 • SPARK - 036/46 52 70 • **Hunedoara:** MEMORII TECH DATA SERVICE - 054/23 07 03 • **Iaşi:** RADIX - 032/ 21 24 53 • SERCOM - 032/12 25 89
Piatra Neamţ: SOFT COMPREST - 033/ 22 21 21 • **Ploieşti:** PLATIN SISTEMS - 044/19 18 18 • **Reşiţa:** ELCOM - 055/22 07 11 • **Suceava:** PRONOVA SMART - 030/52 14 85
Timişoara: ETA–2U: 056/19 95 08 • NETRING COMPUTER - 056/13 28 94 • **Tg. Mureş:** ELECTRO ORIZONT - 065/16 65 72 • **Tulcea:** MICROLINE - 040/51 63 69
CENTRU DE SERVICE AUTORIZAT: 330 73 00

Aprilie 1998
Vol. 8/3

Lei 5000

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

DIN NOU, ARITMETICĂ

Fractali și sisteme de numerație

Numere FIBONACCI

PROVOCĂRI

Un algoritm de factorizare

TURNURILE din Hanoi în Java

REALITATEA MILLENNIUM BUG

SOLUȚII

OLIMPIADA națională ORADEA 1998

PORTRET

Povestea DOS-ului

PLUS

Memorie inteligentă

Probleme propuse

și altele

T.P.
Aprobat R.A.P.R.
Nr.103/DC/053/1998
Valabil - 1998 -

INFADRES

aveți
• un domeniu de activitate
• mai multe birouri/sedii
• angajați
• multe hârtii și probleme de rezolvat
HARABABURĂ !!!
vă doriți
• echipamente hardware interconectate
• software de interfață & de aplicație
• școlarizarea angajaților
ORDINE !!!
vă oferim
• soluții hardware & software
• sisteme integrate "la cheie"
• asistență tehnică software & service hardware
• consultanță
• training
cartea noastră de vizită
• System Integration & Project Management - proiecte la nivel național
• IBM Business Partner
• Microsoft Authorized Distribution Partner/ DSP for Romania & Microsoft Hot Line
• distribuitor Symantec
• distribuitor de PC-uri FORTE
• 9 sucursale FORTE
Microsoft
WHERE DO YOU WANT TO GO TODAY ?™
Authorized Distribution Partner/
DSP for Romania
FORTE
COMPUTERS
Flexibility Made in Romania
IBM
Lipscani 102, București
Tel: 3122360/6137282 Fax: 3122630 Microsoft Hot Line: 3120948

Vol. 8, Nr. 3, Aprilie 1998

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu* (București)
Redactor șef:
prof. *Clara Ionescu* (Cluj)
Editor executiv:
Mircea Sârbu (Tg-Mureș)
Au mai colaborat:
Constantin Săplăcan (Cluj-Napoca)
Gabriel Proșcanu (București)
Florin Sasu (București)
Stelian Ciurea (Sibiu)
Emanuela Mateescu (Iași)
Doru Popescu Anastasiu
Horia Giurgiu (Cluj-Napoca)

Tehnoredactare:
Călin Borz
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
C.P. 172-1, 4300 Tg-Mureș
Adresa e-mail: office@ginfo.ro
Tel: (065) 166.516
Fax: (065) 166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *S.C. Infopress srl*,
Odorheiu Secuiesc,
tel: 066.216.483

Editor: Computer Press Agora srl
str. T. Vladimirescu nr. 63/1
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director editorial:
Mircea Sârbu
Director tehnic:
Szabó László
Reprezentanța București:
Director: *Dana Pastramă*
Tel: (01) 330.92.81

Sumar

soluții.. PAG. 4

probleme REZOLVATE
PAGINI REALIZATE DE CLARA IONESCU
Probleme de la Olimpiada Națională, Oradea,1998.

probleme.. PAG. 21

propuse pentru REZOLVARE
***
Probleme de la concursul ACM și de la antrenamentul lotului național.

provocări.. PAG. 41

Un algoritm de factorizare
DE CONSTANTIN SĂPLĂCAN
O propunere care trebuie verificată.

Turnurile din Hanoi în Java
DE FLORIN SASU
O rezolvare iterativă, deși mai complicată, oferă performanță.

focus.. PAG. 26

Fractali și sisteme de numerație
DE STELIAN CIUREA
Fascinația formelor și a aritmeticii își dau mâna într-un domeniu cu aplicații diverse.

frontiere.. PAG. 49

IRAM sau memoria inteligentă
DE HORIA GIURGIU
Memoria calculatoarelor va fi controlată de procesoare speciale, pentru un acces mai rapid.

redactorul.. PAG. 52

Millennium Bug
DE MIRCEA SÂRBU
Ce se va întâmpla în anul 2000? Un apocalips birocratic, spun pesimiștii.

mate.. PAG. 31

Numere Fibonacci
DE EMANUELA MATEESCU
Lucrările unui matematician din secolul XII sunt încă actuale.

Principiul inducției în informatică
DORU POPESCU ANASTASIU
Se poate folosi inducția în programe?

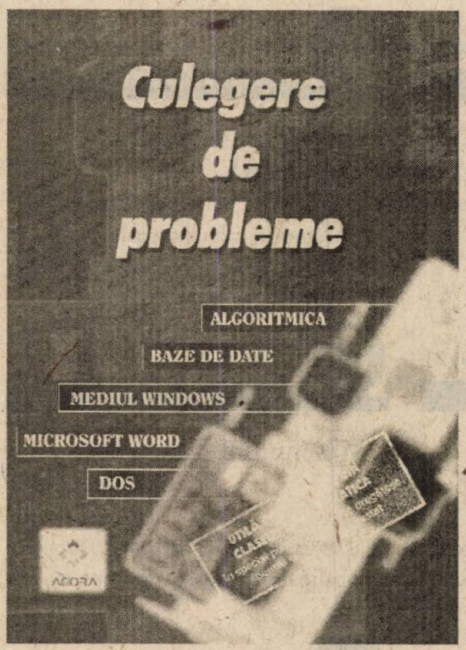
profil.. PAG. 47

Povestea DOS-ului
DE GABRIEL PROȘCANU
De ce a echipat IBM primele PC-uri cu sistemul de operare de la Microsoft? Nimeni nu știe precis...

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Culegerea conține:

- probleme de algoritmică, majoritatea rezolvate în Pascal
- probleme de gestiune a bazelor de date (FOX-PRO)
- probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word
- probleme de verificare a cunoștințelor aferente sistemului de operare DOS

Această culegere constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru atestatul de informatică dar poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, culegerea conține o varietate mare de enunțuri de probleme rezolvate, abordând diferite metode și stiluri de programare.

• • •

Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) mai elegant și cu o eficiență în programare mai ridicată.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422] sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Olimpiada de PRIMĂVARĂ

În afară de faptul că a venit primăvara, ce altceva ar putea constitui tema unui editorial? Cea mai cuminte variantă este să încerce să prezinte în mod sintetic numărul pe care îl prefațează. Lipsit de imaginație și de inspirație cum s-a nimerit să fiu azi, am optat pentru această variantă.

ONI '98

Dar... Cum aș putea să nu vorbesc despre Olimpiada Națională de Informatică? Mai ales că secțiunea **Rezolvări** din acest număr este mult mai substanțială decât de obicei, tocmai pentru a putea prezenta cât mai multe dintre soluțiile problemelor din concurs...

Deci, pentru prima oară în viață, am "participat" la ONI. Anul acesta, concursul a fost găzduit de Liceul "Emanuil Gojdu" din Oradea. Adevărul este că am fost impresionat de eforturile organizatorilor (aici numele profesorilor Adrian și Maria Niță se cuvin amintite), ale Comisiei Centrale (cu riscul de a nedreptăți pe mulți, voi aminti numele d-rei Anca Voicu) și, bineînțeles, ale concurenților.

Am evaluat posibilitatea de a consacra în întregime acest număr Olimpiadei de la Oradea, dar am considerat că e mai bine să ne limităm la prezentarea unui număr sporit de probleme rezolvate (mai ales având în vedere gradul de dificultate al problemelor). Totuși, sunt mândru să consemnez faptul că editura Computer Press Agora - prin GInfo - a fost unul dintre sponsorii manifestării, acordând numeroase premii constând în cărți, CD-ROM-uri și abonamente la Gazeta de Informatică și PC Report. Pentru o mulțime de informații suplimentare (inclusiv subiectele și rezolvările) vă recomand să consultați situl web al Liceului Gojdu (http://www.lego.soroscj.ro/).

Am rămas totuși cu câteva nedumeriri... În ce măsură nivelul de performanță al "elitei" școlare românești reflectă un fenomen "de masă"? Cât de potrivită este specializarea excesivă în informatică încă din primii ani de liceu? Cum se explică prezența feminină redusă la fazele înalte ale competiției (am înțeles că, totuși, anul acesta au fost mai multe fete decât în anii precedenți)? Dar, despre toate acestea sper să pot discuta mai pe larg cu noile mele cunoștințe din lumea informaticii școlare...

Noutăți în cuprins

Revenim deci la noul număr al revistei. Veți observa cu siguranță câteva elemente noi. Nu este întâmplător, mai ales în condițiile în care dialogul cu cititorii noștri s-a mai "dezghețat" și am constat cât de diverse pot fi pretențiile la care GInfo trebuie să facă față. Am considerat, deci, că o structură prea rigidă este incomodă, mai ales acum, când căutăm căile cele mai potrivite...

Așadar, au apărut secțiuni noi. **Provocări** cuprinde materiale care prezintă idei incitante, asupra cărora cititorii sunt chemați să se pronunțe. De pildă, algoritmul de factorizare propus în acest număr își caută încă o implementare, iar algoritmul iterativ de rezolvare a problemei turnurilor din Hanoi ar putea fi, eventual, îmbunătățit. Cine încearcă?

Secțiunea **Realitatea** s-a născut dintr-o idee dezbătută mult pe lista "dialog". Programarea reală nu înseamnă doar algoritmică. "Criza anului 2000" este un subiect extrem de actual și se datorează în mare măsură unei viziuni înguste asupra aplicațiilor dezvoltate în anii '60-'70. Tipurile de date sunt și ele implicate și un premiu se oferă...

Ce a rămas?

Aritmetica s-a întâmplat să fie foarte prezentă în acest număr. Mai cu seamă sistemele de numerație. Pe baza acestora se pot realiza programe interesante pentru desenarea fractalilor, așa cum veți afla citind secțiunea **Focus**. Un interesant sistem de numerație i se datorează unui matematician pe nume Fibonacci, personajul principal al unuia dintre articolele secțiunii **Mate**. Tot aici veți afla cum o tehnică matematică utilizată îndeosebi pentru generalizarea unor rezultate experimentale, inducția, poate fi folosită în programare.

Portretul din acest număr este de fapt unul dublu: Gary Kildall și Bill Gates. Care au fost rațiunile pentru care PC-urile au fost echipate cu sistemul de operare DOS și nu cu CP/M? Se pare că nu vă veți lămuri citind "Povestea DOS-ului", dar veți afla detalii interesante și vă veți consola cu ideea că, se pare, nimeni nu știe...

Deși mai veche, secțiunea **Frontiere** marchează un debut: un articol despre hardware. În speță despre îmbinarea procesoarelor cu memoria. Promisiunile acestei tehnologii sunt uriașe.

Sincope

Există și două secțiuni care de data aceasta lipsesc. Este vorba de **Serial** și **Internet**. Nu trebuie să înțelegeți însă că lecțiile de Java s-au terminat. Dl. Prof. Georgescu este deocamdată în străinătate, dar serialul va continua. În acest număr, am încercat să vă oferim totuși "porția" de Java (în secțiunea Provocări). Cât despre articolul despre Internet, el a fost înlocuit în acest număr de secțiunea experimentală "Realitatea". Chiar dacă ecourile acestei încercări sunt pozitive, ea nu va înlocui partea de Internet.

Fructele dialogului

Înainte de a încheia, trebuie să remarc o noutate pe care o așteptam: au apărut primele materiale semnate de elevi. Alături de varianta iterativă a rezolvării problemei turnurilor din Hanoi (datorată lui Florin Sasu), veți găsi și două probleme interesante propuse de elevi. Ne interesează mult părerea voastră despre aceste materiale (chiar dacă problemele propuse nu intră în calculele pentru concursul de programare).

Veți remarca cu siguranță o secțiune **Dialog** extinsă, care reflectă discuțiile de pe lista "dialog". În fine, sper ca la momentul apariției revistei pe piață, situl Web www.ginfo.ro să cuprindă deja integral materialul publicat în numărul 1/98 al revistei.

Mircea Sârbu

Premiul lunii: **Cătălin Drula**

SOLUȚIILE problemelor propuse

Primăvara prevestește olimpiadele școlare... sau olimpiadele prevestesc primăvara? Anul acesta cei mai buni elevi informaticieni s-au întrecut într-o competiție extrem de dificilă, perfect organizată și desfășurată la Oradea. Vom publica problemele și rezolvările lor în câteva numere ale revistei noastre, și vă adresăm o chemare de a le studia și de a căuta să le rezolvați conform stilului propriu, pe baza unor idei originale pentru a vă verifica puterile și talentul.

Clasa a IX-a: Scuturi

Problemă propusă de prof. Rodica Pintea, Liceul "Grigore Moisil", București

Enunț:

Pe o linie orizontală, la distanțe egale, se află `n` obiective punctiforme, numerotate de la `1` la `n`, înzestrate fiecare cu un scut.

Pe o linie paralelă cu aceasta se deplasează, într-o mișcare de "du-te vino" două dispozitive de tragere care încearcă să distrugă obiectivele considerate.

Primul dispozitiv pornește din dreptul obiectivului `1`, se deplasează succesiv în dreptul obiectivelor `2,3,...,n-1,n,n-1,...` etc. Al doilea dispozitiv pornește din dreptul obiectivului `n` și se deplasează în dreptul obiectivelor `n-1,n-2,...,2,1,2,3,...` etc.

Ambele dispozitive parcurg distanța din dreptul unui obiectiv până în dreptul obiectivului următor într-o secundă. Pentru fiecare dispozitiv se cunoaște un număr `p1`, respectiv `p2`, reprezentând numărul de secunde dintre două trageri consecutive. Cele două dispozitive pot să tragă doar asupra obiectivului în dreptul căruia se află. În momentul pornirii, ambele dispozitive trag.

Pentru fiecare obiectiv `i` se cunoaște un număr `ri` (`i<=n`) reprezentând rezistența scutului, acesta însemnând că obiectivul `i` este distrus după `ri+1` trageri asupra sa.

Observații:

- dispozitivele se deplasează și trag independent unul de celălalt (astfel încât ele pot trage asupra aceluiași obiectiv în același moment; în acest caz scutul este atacat de două ori);
- timpul unei trageri este neglijabil;
- un dispozitiv continuă să tragă și asupra obiectivelor deja distruse.

Să se afișeze pe ecran:

a) numărul maxim de obiective care pot fi distruse, considerând mișcarea dispozitivelor și resurselor de muniție infinite;

b) timpul minim după care se reușește distrugerea obiectivelor numărate la punctul a).

Date de intrare:

Datele de intrare se citesc din fișierul `SCUT.IN` având următorul format:

- pe prima linie este scris numărul întreg `n`, reprezentând numărul de obiective (`2≤n≤30`);
- pe linia a doua se află `n` numere naturale având cel mult 6 cifre, reprezentând rezistența scuturilor;
- pe a treia linie se găsesc două numere naturale reprezentând perioada de tragere a primului, respectiv a celui de-al doilea dispozitiv (`0<p1,p2≤n`).

Exemplu:

SCUT.IN	Pe ecran se vor afișa numerele:
5	3
2 1 8 13 2	34
2 4	

Observații:

- pentru exemplul de mai sus obiectivele distruse sunt cele cu numerele de ordine: `1`, `3` și `5`;

Timp maxim de executare/test: 3 secunde.

Rezolvare:

Pentru această problemă există o rezolvare banală prin

simularea mișcării și decrementarea rezistenței scuturilor până la distrugerea acestora. Din păcate o asemenea simulare este mare consumatoare de timp în executare.

În scopul de a obține un timp de executare mai bun, se simulează câte o "perioadă" completă de tragere pentru fiecare dispozitiv (până când acesta trage din nou, aflându-se în poziția inițială). Obiectivele atinse în fiecare dintre "perioadele" ambelor dispozitive sunt cele ce vor fi, in final, distruse.

Se calculează o "perioadă" comună `k` (după care evoluția sistemului se va relua) ca fiind cel mai mic multiplu comun al valorilor "perioadelor" celor două dispozitive. Se mai calculează numărul de atacuri `c(i)` executat asupra fiecărui scut `i` de ambele dispozitive într-o astfel de "perioadă comună".

Numărul de "perioade comune" necesare este dat de cel mai mare dintre rapoartele `r(i)/a(i)`. Se consideră acest număr de perioade mai puțin una (calculându-se direct timpul necesar), ultima perioadă urmând a fi simulată, deoarece ea nu este (decât în cazurile particulare) o perioadă completă.

Listing SCUTURI.PAS

```
Program scuturi;
const nmax=30;
type scut=array[1..nmax] of Longint;
var
    n,i,j,p,k,max,timp:Longint;
    r:scut;
    per,poz,pp,sens:array[1..2] of Longint;
    a:array[1..2,1..2*nmax] of 1..2*nmax;
    b:array[1..2] of Longint;
    c:array[1..2*nmax] of Longint;
    valid:Boolean; f:Text;

  function gata:Boolean;
  var i,j:Word;
  begin
    gata:=false;
    for i:=1 to 2 do
      for j:=1 to n do
        if (r[j]>0) and (a[i,j]>0) then Exit;
    gata:=true
  end;

  function cmmdc(a,b:Longint):Longint;
  var r:Longint;
  begin
  repeat
      r:=a mod b;
      a:=b; b:=r
    until r=0;
    cmmdc:=a
  end;

Begin
  Assign(f,'SCUT.IN'); Reset(f); Readln(f,n);
  poz[1]:=1; poz[2]:=n; Read(f,r[1]);
  for i:=2 to n-1 do
    begin Read(f,r[i]); Inc(r[i]) end;
  Read(f,r[n]);
  Readln(f,per[1],per[2]);
  { se determina perioada principala          }
  { a fiecarui tragator                       }
  sens[1]:=1; sens[2]:=-1;
  for i:=1 to 2 do
  begin
    p:=poz[i];
    repeat
      Inc(a[i,p]);
      for j:=1 to per[i] do
      begin
        if (p=n) and (sens[i]=1) then sens[i]:=-1;
        if (p=1) and (sens[i]=-1) then sens[i]:=1;
        p:=p+sens[i]
      end;
      Inc(b[i],per[i])
    until p=poz[i]
  end;
    { se determina obiectivele distructibile (!) }
  k:=0;
  for i:=1 to n do
    if a[1,i]+a[2,i]>0 then Inc(k);
  Writeln; Writeln(k);
    { se determina perioada ansamblului         }
  k:=b[1]*b[2] div cmmdc(b[1],b[2]);
  for j:=1 to n do
  begin
    c[j]:=0;
    for i:=1 to 2 do
      c[j]:=c[j]+a[i,j]*k div b[i]
  end;
    { se determina numarul de perioade necesare }
  max:=0;
  for j:=1 to n do
    if (c[j]>0) and (r[j] div c[j]>max)
    then
      begin max:=r[j] div c[j]; i:=j end;
  {  se distrug scuturile corespunzatoare       }
  {  numarului de perioade -1 pentru ca ultima  }
  {  perioada nu se executa neaparat complet    }
  timp:=timp+(max-1)*k;
  for i:=1 to n do
  begin
    j:=r[i]-(max-1)*c[i];
    if j>0 then r[i]:=j
           else r[i]:=0
  end;
     { se simuleaza ultima perioada             }
  sens[1]:=1; sens[2]:=-1;
  repeat
```

```
    for i:=1 to 2 do
    begin
      if (poz[i]=n) and (sens[i]=1)
      then sens[i]:=-1;
      if (poz[i]=1) and (sens[i]=-1)
      then sens[i]:=1;
      poz[i]:=poz[i]+sens[i];
      pp[i]:=(pp[i]+1) mod per[i];
      if (pp[i]=0) and (r[poz[i]]>0)
      then Dec(r[poz[i]])
    end;
    Inc(timp)
  until gata;
  Writeln(timp)
End.
```

Clasa a IX-a: Fracții

Problemă propusă de conf. dr. Henri Luchian, Universitatea "A. I. Cuza", Iași

Enunț:

Se consideră primele N (1≤N≤26) litere mari din alfabetul englezesc care reprezintă nume de variabile. Aceste litere se scriu succesiv în ordine alfabetică, delimitate prin operatorul de împărțire ':', formând o expresie algebrică. Prin eventuala adăugare la această expresie a unor perechi de paranteze rotunde, se obține o fracție etajată. Fracția poate fi scrisă, în urma eliminării etajelor, sub forma unei fracții simple (care are la numărător, respectiv numitor, câte un produs de variabile). Cunoscând setul de litere care formează produsul de la numărător, se cere să se indice o modalitate de a așeza parantezele.

Date de intrare:

Fișierul text FRACTIE.IN conține două linii. Pe prima linie se află N (numărul de litere folosite), iar pe a doua se află un șir de litere mari, distincte, ordonate alfabetic, fără spații, reprezentând variabilele de la numărător. Datele de intrare sunt corecte.

Date de ieșire:

Fișierul text FRACTIE.OUT conține pe o singură linie expresia cu paranteze fără nici un spațiu.

Observații:

- Expresia trebuie să fie corect parantezată.
- Dacă nu există soluție, fișierul de ieșire va fi format dintr-o singură linie conținând mesajul: NU
- Dacă există mai multe soluții, se va scrie una singură.

Exemplu:

FRACTIE.IN	FRACTIE.OUT
5	A:(B:(C:D):E)
ACE	

Timp maxim de executare/test: 5 secunde.

Rezolvare:

Este o problemă nestandard, inspirată de o problemă dată la o olimpiadă rusă de matematică. Pentru a găsi ideea de rezolvare, se poate căuta o regulă de ... ne-asociativitate, astfel ca, parcurgând o expresie cu paranteze, să se decidă de care parte a liniei de fracție se va afla fiecare variabilă; se observă efectul unei paranteze închise/deschise asupra a două variabile succesive, apoi efectul parantezelor multiple. Fiecare variabilă poate ajunge fie la numărător, fie la numitor, în funcție de modul în care se pun parantezele; singurele excepții sunt x1, care nu poate apărea decât la numărător, și x2, care nu poate apărea decât la numitor (este ușor de demonstrat - dar problema noastră nu cere acest lucru - că numărul de fracții diferite ce se pot obține este 2^(n-2), n fiind numărul de variabile.

Desigur, o fracție din cele 2^(n-2) poate fi obținută prin mai multe feluri de a pune parantezele "( )". O idee de rezolvare poate fi scrierea unui șir de n biți, bitul i fiind 1 dacă x_i trebuie să apară la numitor și 0 dacă x_i trebuie să apară la numărător. Se parcurge șirul, deschizând și închizând paranteze conform observațiilor rezultate mai sus, iar apoi se adaugă perechile eventualelor paranteze incorect deschise/închise la prima trecere, și anume pe nivelul pe care ele apar.

Implementarea pe care v-o prezentăm a fost realizată de *Cătălin Frâncu*, membru în subcomisia clasei a IX-a.

Listing FRACTII.PAS

```
Program fractii;
type Vector=array['A'..'Z'] of Boolean;
var Sus:Vector;
{ Sus[X]=True daca litera X este la numarator     }
    Last:Char;  { Ultima litera                   }

  procedure ReadData;
  var i:Integer;
      C:Char;
  begin
    for C:='A' to 'Z' do Sus[C]:=False;
    Assign(Input,'FRACTIE.IN');
    Reset(Input);
    ReadLn(i);
    Last:=Chr(Ord('A')+i-1);
    while not Eoln do
    begin Read(C); Sus[C]:=True end;
    Close(Input)
  end;

  procedure Parant;
  var Curent:Vector;
      Open,i:Integer;      { Nr. de ( }
      C,C2:Char;
  begin
    Curent['A']:=True;
    for C:='B' to Last do Curent[C]:=False;
```

```
    Write('A:');
    Open:=0;
    for C:='C' to Last do
    begin
      if Sus[C]<>Curent[C]
      then
        begin
          Write('(');
          for C2:=C to Last do
            Curent[C2]:=not Curent[C2];
          Inc(Open)
        end;
      Write(Pred(C), ':')
    end;
    Write(Last);
    for i:=1 to Open do
      Write(')');
    WriteLn
  end;

Begin
  ReadData;
  Assign(Output,'FRACTIE.OUT'); Rewrite(Output);
  if Last='A'
  then
    if Sus['A']
    then WriteLn('A')
    else WriteLn('NU')
  else
    if (not Sus['A']) or Sus['B']
    then WriteLn('NU')
    else Parant;
  Close(Output)
End.
```

Clasa a X-a: La monetărie

Problemă propusă de conf. dr. Radu Mârșanu, Academia de Studii Economice, București

Enunț:

Într-un depozit al monetăriei statului sosesc n saci cu monezi. Șeful depozitului cunoaște numărul de monezi din fiecare sac și ar vrea să modifice conținutul sacilor, prin mutări de monezi dintr-un sac în altul, astfel încât în final, în fiecare sac să fie același număr de monezi. Ajutați șeful depozitului să obțină același număr de monezi în fiecare sac, prin efectuarea unui număr minim de mutări.

Date de intrare:

În fișierul Text MONEZI.IN se va scrie pe prima linie un număr întreg n ($2 \le n \le 2000$), reprezentând numărul de saci. Pe următoarele n linii sunt scrise numere întregi, reprezentând numerele de monezi din fiecare sac (numărul total de monezi din toți sacii $\le 1.000.000.000$).

Date de ieșire:

Rezultatele se vor scrie în fișierul text MONEZI.OUT sub următoarea formă:

- pe prima linie se scrie un număr întreg m, reprezentând numărul minim de mutări necesare;
- pe următoarele m linii se vor scrie triplete de numere întregi:

a b c

unde:

a reprezintă numărul de ordine al sacului din care se mută monezi;

b reprezintă numărul de ordine al sacului în care se mută monezi;

c reprezintă numărul de monezi care se mută din sacul a în sacul b.

Observație:

În cazul în care problema nu are soluție, se va scrie în fișier cuvântul: 'NU'.

Exemplu:

MONEZI.IN	MONEZI.OUT
3	2
35	2 1 5
48	2 3 3
37	

Timp maxim de executare/test: 8 secunde.

Rezolvare:

Subcomisia clasei a X-a a propus un algoritm *greedy* pe care îl prezentăm într-o variantă ușor îmbunătățită.

Facem următoarele observații:

1) Dacă media aritmetică a numerelor de monede din saci nu este număr întreg, problema nu are soluție.

2) În cazul în care există soluție, se vor "pune deoparte" sacii care conțin exact atâtea monede cât este media aritmetică.

3) Cu o mutare se pot "rezolva" cel mult doi saci.

4) Cum nu este indiferentă ordinea în care se "rezolvă" câte un sac, prima dată se vor căuta perechi de saci pentru care există o mutare care "rezolvă" amândoi sacii. Aceștia, de asemenea se consideră "rezolvați" și astfel numărul de saci din nou este posibil să se micșoreze.

În cele ce urmează, se ordonează sacii în ordine descrescătoare după numărul de monede. Se mută atâtea monede din primul sac în ultimul câte sunt necesare pentru a putea pune deoparte ultimul sac. Deoarece, în funcție de numărul de monede rămase în primul sac, este posibil ca acesta să poată forma pereche cu un alt sac, se reia pasul descris la punctul 2) din observație pentru acesta și restul sacilor. În cazul în care nu se găsește un asemenea sac, acestuia trebuie să i se găsească noul loc în șirul ordonat după numărul de monede.

Listing MONEDE.PAS

```
Program monede;
const max=3000;
type sir=array[1..max] of Integer ;
var a,p:sir;
    mut:array[1..max,1..3] of Integer ;
    sf,incep,n,nrmut,i,j,pz,k:Word;
    f:Text; med:Longint;

  procedure Sort(l,r:Word);
  var i,j,x,y:Word;
  begin
    i:=l; j:=r;
    x:=a[(l+r) div 2];
    repeat
      while a[i]>x do i:=i+1;
      while x>a[j] do j:=j-1;
      if i<=j
      then
        begin
          y:=a[i]; a[i]:=a[j]; a[j]:=y;
          y:=p[i]; p[i]:=p[j]; p[j]:=y;
          Inc(i); Dec(j)
        end
    until i>j;
    if l<j then Sort(l,j);
    if i<r then Sort(i,r)
  end;

  procedure pun_deoparte(var n:Word; var a,p:sir);
  var i,pz:Word;
  begin
    pz:=0;
    for i:=1 to n do
      if a[i]<>med
      then
        begin
          Inc(pz); a[pz]:=a[i]; p[pz]:=p[i]
        end;
    n:=pz
  end;

  procedure citire;
  var i,pz:Word;
  begin
    Assign(f,'MONEZI.IN'); Reset(f); Readln(f,n);
    med:=0;
    for i:=1 to n do
    begin Read(f,a[i]); med:=med+a[i]; p[i]:=i end;
    if med mod n <> 0
    then
      begin
        Close(f);
        Assign(f,'MONEZI.OUT'); Rewrite(f);
        Writeln(f,'NU'); Close(f); Halt
      end;
    med:=med div n;
    pun_deoparte(n,a,p);
    Close(f)
  end;

  procedure tiparire;
  var i:Word;
  begin
    Assign(f,'MONEZI.OUT'); Rewrite(f);
    Writeln(f,nrmut);
    for i:=1 to nrmut do
      Writeln(f,mut[i,1],' ',mut[i,2],' ',
                                mut[i,3]);
    Close(f)
  end;

Begin
  citire;
  incep:=1;
  sf:=n;
  nrmut:=0;
  sort(incep,sf);
  i:=incep;
  while a[i]>med do
  begin
    j:=sf;
    while a[j]<med do
    begin
      if a[i]+a[j]=2*med
      then
        begin
          Inc(nrmut);
          mut[nrmut,1]:=p[i]; mut[nrmut,2]:=p[j];
          mut[nrmut,3]:=med-a[j];
          a[i]:=med; a[j]:=med
        end;
      Dec(j)
    end;
    Inc(i)
  end;
  pun_deoparte(n,a,p);
  sf:=n;
  while sf>incep do
  begin
    Inc(nrmut);
    a[incep]:=a[incep]-med+a[sf];
    mut[nrmut,1]:=p[incep]; mut[nrmut,2]:=p[sf];
    mut[nrmut,3]:=med-a[sf];
    Dec(sf);
    if a[incep]=med
    then Inc(incep)
    else
      begin
        i:=incep+1;
        while i<=sf do
        begin
```

```
          if a[incep]+a[i]=2*med
          then
            begin
              k:=i;
              if a[incep]>med
              then j:=incep
              else begin j:=i; i:=incep end;
              Inc(nrmut);
              mut[nrmut,1]:=p[j];
              mut[nrmut,2]:=p[i];
              mut[nrmut,3]:=med-a[i];
              for i:=k+1 to sf do
              begin a[i-1]:=a[i]; p[i-1]:=p[i] end;
              Dec(sf); Inc(incep);
              i:=sf
            end;
          Inc(i)
        end
      end;
      i:=incep+1;
      while (a[i-1]<a[i]) and (i<=sf) do
      begin
        k:=a[i-1]; a[i-1]:=a[i]; a[i]:=k;
        k:=p[i-1]; p[i-1]:=p[i]; p[i]:=k;
        Inc(i)
      end
    end;
  tiparire
End.
```

Clasa a X-a: La monetărie... în ziua a doua

Problemă propusă de prof. Clara Ionescu, Liceul de Informatică, Cluj

Enunț:

Șeful depozitului are trei saci cu monede, niciunul gol. Prietenul lui îi cere un sac gol, deci șeful depozitului va trebui să elibereze unul din saci. Dorind să nu se încurce în evidența monedelor din saci, va muta dintr-un sac în altul un număr de monede egal cu numărul de monede aflate în sacul în care face mutarea. Sacii sunt suficient de mari pentru orice astfel de mutare. Cum va proceda pentru a goli un sac efectuând un număr minim de mutări?

Date de intrare:

Fișierul de intrare `SACI.IN` are o singură linie de forma:

`x y z`

unde `x, y, z` reprezintă numărul de monede din fiecare sac ($1 \le x, y, z < 10000$ și $x+y+z \le 10000$);

Date de ieșire:

Fișierul de ieșire `SACI.OUT` va conține pe prima linie numărul `n` reprezentând numărul de mutări, iar pe următoarele `n` linii se vor scrie două numere naturale:

`a b`

unde:

- `a` reprezintă numărul de ordine al sacului din care se face mutarea;
- `b` reprezintă numărul de ordine al sacului în care se face mutarea.

Observație:

Datele de intrare sunt corecte, nu necesită validare. Problema are soluție întotdeauna.

Exemplu:

SACI.IN	SACI.OUT
5 7 3	3
	1 3
	3 1
	1 3

Timp maxim de executare/test: 3 secunde.

Rezolvare:

Rezolvarea problemei propusă de comisie - implementată de *Valentin Gheorghiță* se bazează pe implementarea metodei *Branch and Bound* pentru fiecare stare posibilă a conținutului sacilor. Se observă imediat că totdeauna există un sac din care se pot muta monede în doi saci, există unul din care se pot muta monede într-unul și există unul din care nu se pot muta monede respectând restricțiile problemei. Se vor genera la fiecare pas toate mutările posibile și se va păstra de fiecare dată adresa stării predecesoare pentru a putea reconstitui "drumul" care conduce la rezultatul căutat.

Listing SACI.PAS

```
Program saci;
type ref=^inr;
     inr=record
           dr,tata:ref;
           a,b,c:Integer
         end;
var a,b,c,pz:Integer;
    f:Text;
    final,temp,rad,poz,caut:ref;
    m:array[1..300,1..3] of Integer;

  function minim(a,b,c,pz:Integer):Integer;
  var t:Integer;
  begin
    if a>b then begin t:=a; a:=b; b:=t end;
    if a>c then begin t:=a; a:=c; c:=t end;
    if b>c then begin t:=c; c:=b; b:=t end;
    case pz of
      1 : minim:=a;
      2 : minim:=b;
      3 : minim:=c
    end
  end;
```

```
procedure initializare;
begin
  New(rad); New(temp);
  rad^.dr:=temp;
  temp^.dr:=nil; temp^.tata:=nil;
  temp^.a:=minim(a,b,c,1);
  temp^.b:=minim(a,b,c,2);
  temp^.c:=minim(a,b,c,3);
  poz:=rad; final:=temp
end;

procedure citire;
begin
  Assign(f,'SACI.IN');
  Reset(f);
  Readln(f,a,b,c);
  Close(f)
end;

function verif:Boolean;
var test:Boolean;
begin
  test:=false;
  if final^.a=final^.b
  then
    begin
      test:=true; m[1,1]:=0;
      m[1,2]:=b+a; m[1,3]:=c; pz:=1
    end;
  if final^.a=final^.c
  then
    begin
      test:=true; m[1,1]:=0;
      m[1,2]:=b; m[1,3]:=c+a; pz:=1
    end;
  if final^.b=final^.c
  then
    begin
      test:=true; m[1,1]:=a;
      m[1,2]:=b+b; m[1,3]:=0; pz:=1
    end;
  verif:=test
end;

procedure cautare;
label 10;
var test:Boolean;
begin
  if verif then goto 10;
  repeat
    poz:=poz^.dr;
    a:=minim(poz^.a+poz^.a,
    poz^.b-poz^.a,poz^.c,1);
    b:=minim(poz^.a+poz^.a,
    poz^.b-poz^.a,poz^.c,2);
    c:=minim(poz^.a+poz^.a,
    poz^.b-poz^.a,poz^.c,3);
    test:=true; temp:=rad;
    repeat
      temp:=temp^.dr;
      if (temp^.a=a) and (temp^.b=b) and
      (temp^.c=c)
      then test:=false
    until (temp^.dr=nil) or not test;
    if test
    then
      begin
        New(temp); temp^.tata:=poz;
        temp^.a:=a; temp^.b:=b; temp^.c:=c;
        temp^.dr:=nil; final^.dr:=temp;
        final:=temp;
        if verif then goto 10
      end;
    a:=minim(poz^.a+poz^.a,
    poz^.b,poz^.c-poz^.a,1);
    b:=minim(poz^.a+poz^.a,
    poz^.b,poz^.c-poz^.a,2);
    c:=minim(poz^.a+poz^.a,
    poz^.b,poz^.c-poz^.a,3);
    test:=true; temp:=rad;
    repeat
      temp:=temp^.dr;
      if (temp^.a=a) and (temp^.b=b) and
      (temp^.c=c)
      then test:=false
    until (temp^.dr=nil) or not test;
    if test
    then
      begin
        New(temp); temp^.tata:=poz;
        temp^.a:=a; temp^.b:=b; temp^.c:=c;
        temp^.dr:=nil; final^.dr:=temp;
        final:=temp;
        if verif then goto 10
      end;
    a:=minim(poz^.a,poz^.b+poz^.b,
    poz^.c-poz^.b,1);
    b:=minim(poz^.a,poz^.b+poz^.b,
    poz^.c-poz^.b,2);
    c:=minim(poz^.a,poz^.b+poz^.b,
    poz^.c-poz^.b,3);
    test:=true;
    temp:=rad;
    repeat
      temp:=temp^.dr;
      if (temp^.a=a) and (temp^.b=b) and
        (temp^.c=c)
      then test:=false
    until (temp^.dr=nil) or not test;
    if test
    then
```

```
        begin
          New(temp); temp^.tata:=poz;
          temp^.a:=a; temp^.b:=b; temp^.c:=c;
          temp^.dr:=nil; final^.dr:=temp;
          final:=temp;
          if verif then goto 10
        end
    until false;
    10:
end;

procedure sfarsit;
begin
  while final<>nil do
  begin
    pz:=pz+1; m[pz,1]:=final^.a;
    m[pz,2]:=final^.b; m[pz,3]:=final^.c;
    final:=final^.tata
  end
end;

function testare(var a,b,c:Integer;
       a1,b1,c1:Integer; m1,m2:Integer):Boolean;
var test:Boolean;
begin
  test:=true;
  if a<>a1 then test:=false;
  if b1<>b+b then test:=false;
  if c1<>c-b then test:=false;
  if test
  then
    begin
      a:=a1; b:=b1; c:=c1;
      Writeln(f,m1,' ',m2)
    end;
  testare:=test
end;

function ct(var a,b,c:Integer;
          a1,b1,c1,m1,m2:Integer):Boolean;
var test:Boolean;
begin
  test:=false;
  if testare(a,b,c,a1,b1,c1,m1,m2)
  then test:=true
  else
    if testare(a,b,c,b1,a1,c1,m1,m2)
    then test:=true
    else
      if testare(a,b,c,c1,a1,b1,m1,m2)
      then test:=true
      else
        if testare(a,b,c,a1,c1,b1,m1,m2)
        then test:=true
        else
          if testare(a,b,c,b1,c1,a1,m1,m2)
          then test:=true
          else
            if testare(a,b,c,c1,b1,a1,m1,m2)
            then test:=true;
  ct:=test
end;

procedure tiparire;
var i:Word;
begin
  Assign(f,'SACI.IN'); Reset(f); Readln(f,a,b,c);
  Close(f);
  Assign(f,'SACI.OUT'); Rewrite(f);
  Writeln(f,pz-1);
  for i:=pz-1 downto 1 do
  begin
    if ct(a,b,c,m[i,1],m[i,2],m[i,3],3,2)
    then
    else
      if ct(a,c,b,m[i,1],m[i,2],m[i,3],2,3)
      then
      else if ct(b,a,c,m[i,1],m[i,2],m[i,3],3,1)
      then
      else
        if ct(b,c,a,m[i,1],m[i,2],m[i,3],1,3)
        then
        else
          if ct(c,a,b,m[i,1],m[i,2],m[i,3],2,1)
          then
          else
           if ct(c,b,a,m[i,1],m[i,2],m[i,3],1,2)
           then
  end;
  Close(f)
end;

procedure stergere;
begin
  temp:=rad;
  while temp<>nil do
  begin
    final:=temp; temp:=temp^.dr; Dispose(final)
  end
end;

Begin
  citire;
  initializare;
  cautare;
  sfarsit;
  tiparire;
  stergere
End.
```

Clasa a XI-a: Un Excel de jucărie

Enunț:

O foaie de calcul (*spreadsheet*) este un tablou dreptunghiular de celule rectangulare. Fiecare celulă poate conține date sau expresii, care pot fi evaluate obținându-se date. Liniile din foaia de calcul sunt numerotate, începând cu `0`, iar coloanele se denumesc cu litere majuscule ale alfabetului englez (primele 26 de linii), apoi folosind combinații de litere (`AA`, `AB` etc). O celulă poate fi referită specificând coloana și linia corespunzătoare (de exemplu, prima celulă este `A0`).

O *foaie de calcul de jucărie* conține cel mult `26` de coloane (denumite de la `A` la `Z`) și cel mult `10` linii (numerotate de la `0` la `9`). Fiecare celulă poate conține date de tip întreg sau expresii aritmetice în care se folosesc doar operatorii binari `+,-,*,/`, cu semnificația de adunare, scădere, înmulțire, respectiv împărțire întreagă. Operanzii pot fi valori întregi sau referințe de celule.

Problema constă în a evalua, dacă este posibil, o foaie de calcul de jucărie dată. Prin evaluarea unei foi de calcul înțelegem înlocuirea tuturor expresiilor cu valorile lor.

Date de intrare:

Pe prima linie a fișierului de intrare `EXCEL.IN` sunt scrise două numere întregi `n` (reprezentând numărul de linii din foaia de calcul) și `m` (reprezentând numărul de coloane), separate prin spațiu.

Fișierul conține în continuare `n*m` linii de date, câte una pentru fiecare celulă. Conținutul celulelor (valoarea sau expresia) este specificat în ordinea crescătoare a liniilor, iar pe fiecare linie în ordinea alfabetică a coloanelor.

Date de ieșire:

Fișierul de ieșire `EXCEL.OUT` va conține mesajul `EVALUARE IMPOSIBILA` sau foaia de calcul evaluată (`n` linii în ordinea dată, fiecare linie conținând cele `m` valori ale celulelor componente, în ordinea dată a coloanelor, separate prin spațiu).

Observații:

Expresiile pot conține paranteze rotunde.

Expresiile nu conțin spații.

Prioritatea operatorilor este cea uzuală.

Datele și valorile absolute ale expresiilor din foaia de calcul de jucărie sunt numere întregi mai mici decât `2.0E+9`.

Fiecare expresie conține cel mult `100` de caractere.

Expresiile date în foaia de calcul de jucărie sunt sintactic corecte (din punctul de vedere al sintaxei Pascal/C).

Exemplul 1:

```
EXCEL.IN                EXCEL.OUT
1 2                     EVALUARE IMPOSIBILA
1+B0
2+A0
```

Exemplul 2:

```
EXCEL.IN                EXCEL.OUT
2 2                     1 6
1                       5 18
A1+A0
5
B0*3
```

Timp maxim de executare/test: 0.5 secunde

Rezolvare:

Vom reprezenta o foaie de calcul de jucărie sub forma unei matrice cu `10` linii și `26` de coloane. Fiecare componentă a matricei este un articol având 3 câmpuri:

`data` expresia conținută în celulă, citită din fișierul de intrare;

`x` câmp "*flag*", care poate avea doar valorile `0`, `1` și `2`, cu semnificația:
- expresie neevaluată;
- expresie în curs de evaluare;
- expresie evaluată;

`v` valoarea calculată a expresiei, dacă `x=2`, sau nedefinit în rest.

Unei foi de calcul de jucărie îi putem asocia un digraf astfel:

- fiecare celulă constituie un vârf al digrafului;
- există arc de la un vârf `v1` al digrafului la un alt vârf `v2` dacă și numai dacă în expresia din celula corespunzătoare lui `v1` intervine o referință la celula corespunzătoare lui `v2`.

Evaluarea foii de calcul este posibilă dacă și numai dacă *digraful asociat nu conține circuite*.

Pentru rezolvarea problemei, vom parcurge foaia de calcul în ordinea crescătoare a liniilor, iar pe fiecare linie în ordinea crescătoare a coloanelor. Pentru fiecare celulă neevaluată (câmpul `x` corespunzător setat pe valoarea `0`), vom seta câmpul `x` pe valoarea `1` (celulă în curs de evaluare) și vom apela o funcție de evaluare a expresiei din celulă. La revenirea din funcția de evaluare setăm câmpul `x` pe valoarea `2`. Dacă pe parcursul evaluării unei expresii apare o referință la o celulă a cărei expresie este de asemeni în curs de evaluare, afișăm mesajul `EVALUARE IMPOSIBILA` și întrerupem executarea programului (deci nu este necesară detectarea prealabilă a eventualelor circuite în digraf, aceasta realizându-se pe parcursul evaluării).

Evaluarea unei expresii presupune analizarea sa sintactică, în scopul identificării unităților sintactice elementare ce intervin în definirea noțiunii de expresie aritmetică (*termen*, respectiv *factor*) și evaluarea fiecărei unități sintactice. Evaluarea unei expresii implică evaluarea expresiilor corespunzătoare tuturor celulelor referite în cadrul expresiei (practic, o *parcurgere în postordine* a arborescenței cu rădăcina în celula ce conține expresia de evaluat).

Listing EXCEL.PAS

```
Program Excel;
const NrMaxLin=10;
      LgMax=100;
type Linie=0..NrMaxLin;
     Coloana='A'..'Z';
     Expresie=string[LgMax];
     Celula=record
              data:Expresie;
              x:Byte;
              v:Longint
             end;
     Foaie=array[Linie,Coloana] of Celula;
var F:Foaie;
    n,l:Linie;
    m,i:Byte; cm,c:Char;
    fout:Text; numeout:string;

  procedure Citire;
  var fin:Text; nume:string;
      l:Linie; c:Char;
  begin
    Assign(fin,'EXCEL.IN'); Reset(fin);
    Readln(fin,n,m); cm:=Chr(ord('A')+m-1);
    for l:=0 to n-1 do
      for c:='A' to cm do Readln(fin,F[l,c].data);
    Close(fin)
  end;

  procedure Afisare;
  var l:Linie; c:Char;
  begin
    for l:=0 to n-1 do
    begin
      for c:='A' to cm do Write(fout,F[l,c].v:11);
      Writeln(fout)
    end;
    Close(fout)
  end;

  function EvaluareExpresie(var e:Expresie;
  var i:Byte):Longint; forward;
  var ll:Linie; cc:Char; cod:Integer ; v:Longint;

    procedure op(var e:expresie; i:Byte; ii:Byte);
      begin
        Val(Copy(e,i,ii-i),v,cod)
      end;

    function EvaluareFactor(var e:Expresie;
    var i:Byte):Longint;
    var ii:Byte;
    begin
      if e[i]='('    { se verifica daca factorul }
      then     { este o expresie intre paranteze }
        begin
          Inc(i);          { se trece peste ')' }
         { se evalueza expresia dintre paranteze }
          EvaluareFactor:=EvaluareExpresie(e,i);
          Inc(i)           { se trece peste ')' }
        end
      else          { factorul este un operand }
        if(e[i]>='A') and (e[i]<='Z')
        then { factorul este o referinta de celula }
          begin
            ll:=Ord(e[i+1])-Ord('0'); cc:=e[i];
            if F[ll,cc].x=2
            then EvaluareFactor:=F[ll,cc].v
            else
              if F[ll,cc].x=1
              then
                begin
                  Writeln(fout,
                          'EVALUARE IMPOSIBILA');
                  Close(fout); Halt
                end
              else
                begin
                  ii:=1;
                  EvaluareFactor:=
                EvaluareExpresie(F[ll,cc].data,ii)
                end;
            Inc(i,2)
          end
        else { factorul este o valoare  intreaga }
          begin
            ii:=i+1;
            while(ii<=Length(e)) and (e[ii]>='0')
                 and (e[ii]<='9') do Inc(ii);
            op(e,i,ii);
            EvaluareFactor:=v;
            i:=ii
          end
    end;

    function EvaluareTermen(var e:Expresie;
                            var i:Byte):Longint;
    var f:Longint; op:Char;
    begin
      f:=EvaluareFactor(e,i);
      { se evalueza primul factor din care este  }
      { constituit termenul                      }
      while (i<=Length(e)) and ((e[i]='*')
                              or (e[i]='/')) do
                          { mai urmeaza factori  }
      begin
        op:=e[i];
        i:=i+1;          { se trece peste '*'    }
        if op='*'
          then f:=f * EvaluareFactor(e,i)
          else f:=f div
```

```
      EvaluareFactor(e,i)
      end;
      EvaluareTermen:=f
    end;

    function EvaluareExpresie(var e:Expresie;
                              var i:Byte):Longint;
    var t:Longint; op:Char;
    begin
      t:=EvaluareTermen(e,i);
        { se evalueza primul termen din expresie }
      while (i<Length(e)) and ((e[i]='+')
                           or (e[i]='-')) do
        { mai urmeaza termeni                    }
      begin
        op:=e[i];
        i:=i+1;          { se trece peste operator }
        if op='+'
        then t:=t+EvaluareTermen(e,i)
        else t:=t-EvaluareTermen(e,i)
      end;
      EvaluareExpresie:=t
    end;

Begin                    { program principal      }
  Citire;
  Assign(fout,'EXCEL.OUT'); Rewrite(fout);
  for l:=0 to n-1 do
    for c:='A' to cm do
      if F[l,c].x=0
      then
        begin
          F[l,c].x:=1; i:=1;
          l,c].v:=EvaluareExpresie(F[l,c].data,i);
          F[l,c].x:=2
        end;
  Afisare
End.
```

Clasa a XI-a: Critici

Problemă propusă de conf. dr. Henri Luchian, Universitatea "Al. I. Cuza", Iași

Enunț:

Un număr de c critici literari și-au spus, fiecare, părerea în chestiunea: "Care din cele r romane cu succes de public sunt de fapt, capodopere?" Opțiunile lor au stârnit discuții aprinse în masele largi de cititori. Momentul este speculat de cele p=c **div** 2 posturi de televiziune care decid să organizeze, simultan și separat talk-show-uri având ca invitați câte doi critici. Televiziunile plătesc criticilor exclusivitatea - deci nici un critic nu poate apărea la două posturi diferite - dar cu sume atât de mari, încât nici una nu-și permite să "dețină", dacă poate, decât exact doi critici.

Cum însă sponsorii și clienții la spațiile publicitare ale televiziunilor cer insistent ca înainte și după reclamele lor, în emisiuni să nu mai existe certuri sau momente tensionate, fiecare post de televiziune este obligat să invite critici cu păreri apropiate, dar nu identice, asupra romanelor - capodopere. Condiția este ca pentru oricare doi critici invitați la același post opțiunile să difere pentru exact un roman.

Dându-se c și r, $1 \le c \le 100$, $1 \le r \le 20$, precum și listele de romane considerate capodopere de fiecare critic, ajutați, dând una din soluțiile posibile, cât mai multe posturi de televiziune să realizeze talk-show-uri.

Date de intrare:

Pe prima linie a fișierului CRITICI.IN este scris numărul de critici c și numărul de romane r separate printr-un spațiu. Pe a i+1-a linie (i=1,...,c) este scrisă lista capodoperelor propuse de criticul i, sub forma unui șir de cel mult r numere naturale, diferite între ele, cuprinse între 1 și r și separate prin spațiu.

Datele de intrare sunt corecte.

Date de ieșire:

Pe prima linie a fișierului CRITICI.OUT se va scrie un număr t, reprezentând numărul de posturi de televiziune care pot realiza talk-show-uri. Pe următoarele t linii se vor scrie numerele de ordine ale celor doi critici invitați de câte un post de televiziune dintre cele ce pot organiza talk-show-uri.

Exemplul 1:

CRITICI.IN	CRITICI.OUT
4 4	1
1 3 2	1 3
4 2 1	
1 2	
3 4 1	

Exemplul 2:

CRITICI.IN	CRITICI.OUT
4 4	2
1	1 4
3 1 4	2 3
4 1	
2 1	

Timp maxim de executare/test: 2 secunde.

Rezolvare:

Fie luând direct listele criticilor, fie lucrând cu șiruri de câte r biți (în al i-lea șir, poziția j este 1 dacă criticul i crede că romanul j este capodoperă, altfel poziția j este 0), se creează graful atașat problemei; graful are c noduri și muchie de la nodul k la nodul q dacă criticii k și q diferă prin opinia asupra unui singur roman.

- În cazul modelării cu șiruri de biți, proprietatea se exprimă prin "cel de-al k-lea șir de biți diferă pe o singură poziție de cel de-al q-lea șir de biți".

- În cazul modelării pe liste, proprietatea cerută se exprimă astfel: "fie lista k include lista q (în termenii teoriei mulțimilor) și mai are exact un element în plus, fie lista q include lista k și mai are exact un element în plus".

Problema necesită construirea unui graf în care orice circuit este de lungime pară (de fapt, la un graf bipartit). Cuplajul maximal se poate afla elegant - cu flux, cu drumuri alternate -, euristic - greedy, - sau cu backtracking. Soluția greedy este destul de tare: cazurile în care această metodă nu conduce la rezultat optim se descoperă relativ dificil; astfel de cazuri se pot obține în cazul în care numărul de noduri (c) este relativ mare.

Implementarea realizată de *Dan Podeanu* din București a fost pusă la punct în timpul concursului:

Listing CRITICI.PAS

```
Program critici;
type critic=array[0..20] of Byte;
     critic_list=array[1..100] of critic;
     adiacenta=array[1..100,1..100] of Byte;
     big_adiacenta=array[1..102,1..102] of Byte;
var c,r:Byte;
    cl:critic_list;
    a:adiacenta;
    b,b2:big_adiacenta;
    s,t,n:Byte;

  procedure InitData;
  { Se initializeaza cu 0 matricea cu listele    }
  { de capodopere propuse de critici si matricea }
  { de adiacenta                                 }
  begin
    FillChar(cl,Sizeof(cl),0);
    FillChar(a,Sizeof(a),0)
  end;

  procedure ReadData;
  { Se citesc datele de intrare: numarul de      }
  { critici si pentru fiecare critic lista       }
  { capodoperelor propuse de acesta              }
  var fin:Text; i:Integer;
  begin
    Assign(fin,'CRITICI.IN'); Reset(fin);
    Read(fin,c); Readln(fin,r);
    for i:=1 to c do
    begin
      cl[i][0]:=0;
      while not Eoln(fin) do
      begin
        Inc(cl[i][0]); Read(fin,cl[i][cl[i][0]])
      end;
      Readln(fin)
    end;
    Close(fin)
  end;

  function QueryMatch(x,y:Byte):Boolean;
  { Functia returneaza true daca preferintele    }
  { criticilor x si y difera exact pentru un     }
  { roman si false in caz contrar                }
  type hash=array[1..20] of Byte;
  var ii:Byte;
      diffs:Integer;
      v1,v2:critic;
      h:hash;

    function GetMin(first,second:Byte):Byte;
    begin
      if (first<second) then GetMin:=first
                        else GetMin:=second
    end;

  begin
    if (Abs(cl[x][0]-cl[y][0])<>1)
    then begin QueryMatch:=false; Exit end;
    ii:=1; diffs:=0;
    FillChar(h,Sizeof(h),0);
    if (cl[x][0]<cl[y][0])
    then begin v1:=cl[x]; v2:=cl[y] end
    else begin v1:=cl[y]; v2:=cl[x] end;
    for ii:=1 to v1[0] do h[v1[ii]]:=1;
    for ii:=1 to v2[0] do
      if (h[v2[ii]]=0) then Inc(diffs);
      if (diffs=1) then QueryMatch:=true
                   else QueryMatch:=false
  end;

  procedure MakeAdiacenta;
  { Construieste matricea de adiacenta, varfurile }
  { grafului vor reprezenta criticii, iar intre   }
  { doua varfuri i si j va exista muchie daca     }
  { parerile celor doi critici difera exact       }
  { pentru un roman                               }
  var i,j:Integer;
  begin
    for i:=1 to c do
      for j:=i+1 to c do
        if QueryMatch(i,j)
        then begin a[i,j]:=1; a[j,i]:=1 end
        else begin a[i,j]:=0; a[j,i]:=0 end
  end;

  function GetPos(x:Byte):Byte;
  begin
    GetPos:=x+1
  end;

  function GetS:Byte;
  begin
    GetS:=1
  end;
```

```
function GetT:Byte;
begin
  GetT:=c+2
end;

procedure MakeFluxInitialData;
{ Se determina cuplajul maximal                    }
var i,j:Integer;
begin
  for i:=1 to c do
  begin
    for j:=i+1 to c do
      if a[i,j]=1
      then
        begin
          if cl[i][0] mod 2=1
          then b[GetPos(i),GetPos(j)]:=1
          else b[GetPos(j),GetPos(i)]:=1
        end;
    if cl[i][0] mod 2=1
    then b[GetS,GetPos(i)]:=1
    else b[GetPos(i),GetT]:=1
  end
end;

procedure FluxIt;
type
  reached_set=set of Byte;
  path_array=array[1..256] of Byte;
var reached:reached_set;
    path:path_array;
    atpath:Integer;
    FindT:Boolean;
    i:Integer;
    min:Integer;

  procedure Way(from:Byte; at:Byte);
  var i:Integer;
  begin
    path[at]:=from;
    reached:=reached+[from];
    if from=t
    then begin FindT:=true; atpath:=at; Exit
end;
    for i:=1 to n do
      if (b[from,i]=1) and not(i in reached)
                       and not(FindT)
        then Way(i,at+1)
  end;

begin
  repeat
    atpath:=0;
    reached:=[]; FindT:=false;
    Way(s,atpath +1);
    if FindT
    then
      begin
        min:=32767;
        for i:=1 to atpath-1 do
          if b[path[i],path[i+1]]<min
          then min:=b[path[i],path[i+1]];
        if min=32767
        then
          begin
            Writeln('Error: invalid min at
            flux.');
            Halt
          end;
        for i:=1 to atpath-1 do
        begin
          Dec(b[path[i],path[i+1]],min);
          Inc(b[path[i+1],path[i]],min)
        end
      end
  until not FindT
end;

procedure WriteResult;
{ Scrie solutia in fisier                          }
var i,j:Integer;
    fout:Text;
    count:Integer;
begin
  Assign(fout,'CRITICI.OUT'); Rewrite(fout);
  count:=0;
  for i:=GetPos(1) to GetPos(c) do
    for j:=GetPos(1) to GetPos(c) do
      if (b[i,j]-b2[i,j]=-1) then Inc(count);
  Writeln(fout,count);
  for i:=GetPos(1) to GetPos(c) do
    for j:=GetPos(1) to GetPos(c) do
      if b[i,j]-b2[i,j]=-1
      then Writeln(fout,i-1,' ',j-1);
  Close(fout)
end;

Begin
  InitData;
  ReadData;
  MakeAdiacenta;
  FillChar(b,Sizeof(b),0);
  FillChar(b2,Sizeof(b2),0);
  MakeFluxInitialData;
  b2:=b;
  s:=GetS; t:=GetT; n:=c+2;
  FluxIt;
  WriteResult
End.
```

Clasa a XI-a: Drumul cât mai lung

Problemă propusă de prof. Emil Onea,
Liceul "Unirea", Focșani

Enunț:

Doi îndrăgostiți se află într-un oraș k și au la dispoziție p ore (p≤150), timp în care să se plimbe cu mașina și apoi să ajungă fiecare la el acasă, fata în orașul i, iar băiatul în orașul j.

Ei doresc să se plimbe cât mai mult timp împreună, dar să ajungă în cel mult p ore, fiecare în orașul său. Mașina se deplasează cu viteză constantă (cea legal admisă).

Cei doi au la dispoziție o hartă cu n orașe (3≤n≤200) printre care se află, evident, orașul de plecare k și orașele destinație i și j. Harta indică orașele legate prin șosele și durata exprimată în ore a parcurgerii cu viteză constantă (legal admisă) a distanței dintre două orașe. Să se determine traseul comun pe care îl vor parcurge cei doi îndrăgostiți astfel încât timpul petrecut împreună să fie maxim și să ajungă amândoi la propria destinație în cel mult p ore de la plecare.

Observații:

- cronometrarea duratei plimbării se face începând cu momentul 0 din orașul de plecare k.
- în orice oraș unul dintre ei pot continua drumul separat, cu o altă mașină; (despărțirea nu necesită timp suplimentar);
- pe fiecare șosea se circulă în ambele sensuri;
- după plecarea din orașul k nu se staționează nici un moment;
- nu sunt permise întoarceri pe șosea sau parcurgeri incomplete ale unei șosele.

Date de intrare:

- Pe prima linie a fișierului de intrare MASINA.IN se află două numere naturale n și m despărțite printr-un spațiu, reprezentând numărul de orașe de pe hartă și numărul de șosele ce leagă aceste orașe.
- Pe cea de a doua linie se află două numere naturale k și p, despărțite printr-un spațiu, reprezentând numărul de ordine al orașului de plecare și numărul de ore pe care îl au la dispoziție cei doi îndrăgostiți.
- Pe cea de a treia linie sunt scrise două numere naturale i și j despărțite printr-un spațiu, reprezentând numerele de ordine ale orașelor de destinație pentru cei doi.
- Pe următoarele m linii sunt scrise triplete de numere naturale a b d, despărțite prin câte un spațiu, reprezentând faptul că între orașele a și b există șosea directă, respectiv durata d în ore a parcurgerii acesteia.

Date de ieșire:

Rezultatele se vor scrie în fișierul MASINA.OUT structurate pe două linii:

- pe prima linie este scris numărul natural dmax, reprezentând durata maximă (în ore) în care cei doi se vor plimba împreună;
- pe linia a doua sunt scrise numerele naturale: k x1,x2,..., reprezentând orașele pe care le vor vizita împreună, în ordine, începând cu orașul k de plecare (datele vor fi despărțite prin câte un spațiu).

În cazul în care soluția optimă nu este unică se va afișa una singură.

Exemplu:

MASINA.IN	MASINA.OUT
8 9	6
7 8	7 6 5 4 3
1 2	
1 3 1	
3 4 1	
4 2 1	
4 5 1	
4 6 2	
5 6 3	
6 8 1	
7 8 1	
7 6 1	

Timp maxim de executare/test: 4 secunde.

Listing DRUM.PAS

```
Program drumul_cat_mai_lung;
type sir=array[1..200] of Word;
     ssir=array[1..30000] of record v,t:Byte end;
var a:array[1..200] of ^sir;
    r:array[1..200,0..300] of Byte;
    act,xx:^ssir;
    s:array[1..200] of Boolean;
    d,db,df:array[1..200] of Word;
    t:array[1..200] of Byte;
    i,j,k,kk,q,min,m,n,p,bo,fo,st,w:Word;
    f,g:Text;

  procedure citeste;
  begin
    Assign(f,'MASINA.IN'); Reset(f);
    Readln(f,n,m); Readln(f,st,p); Readln(f,bo,fo);
    for i:=1 to n do
    begin
      New(a[i]);
      for j:=1 to n do a[i]^[j]:=15000
    end;
    for i:=1 to n do a[i]^[i]:=0;
    for k:=1 to m do
      begin
        Readln(f,i,j,a[i]^[j]);
        a[j]^[i]:=a[i]^[j]
      end;
    Close(f)
  end;
```

```
procedure Dijkstra;
begin
  for i:=1 to n do
  begin
    d[i]:=a[w]^[i];
    t[i]:=w
  end;
  Fillchar(s,Sizeof(s),false);
  t[w]:=0; s[w]:=true;
  repeat
    min:=15000; k:=0;
    for i:=1 to n do
      if not s[i] and (d[i]<min)
      then
        begin
        min:=d[i];
        k:=i
        end;
    if k<>0
    then
      begin
        s[k]:=true;
        for i:=1 to n do
          if not s[i] and (d[i]>d[k]+a[k]^[i])
          then
            begin
              d[i]:=d[k]+a[k]^[i];
              t[i]:=k
            end
      end
  until k=0
end;

procedure parcurge;
begin
  New(act); New(xx);
  Fillchar(r,Sizeof(r),255);
  r[st,0]:=0;
  k:=1; act^[1].v:=st; act^[1].t:=0;
  repeat
    q:=0;
    for w:=1 to k do
      for i:=1 to n do
        if (act^[w].t+a[act^[w].v]^[i]<=p) and
          (r[i,act^[w].t+a[act^[w].v]^[i]]=255)
        then
          begin
            r[i,act^[w].t+a[act^[w].v]^[i]]:=
                act^[w].v;
            q:=q+1;
            xx^[q].v:=i;
            xx^[q].t:=act^[w].t+a[act^[w].v]^[i]
          end;
        k:=q; act^:=xx^
  until q=0
end;
```

```
procedure rezolva;
begin
  w:=bo; Dijkstra;
  db:=d; w:=fo; dijkstra;
  df:=d; parcurge
end;

procedure scrieoras(i,k:Byte);
begin
  if i<>0
  then
    begin
      scrieoras(r[i,k],k-a[i]^[r[i,k]]);
      Write(g,i,' ')
    end
end;

procedure solutie(i,k:Byte);
begin
  Writeln(g,k);
  scrieoras(i,k)
end;

procedure scrie;
begin
  Assign(g,'MASINA.OUT');
  Rewrite(g);
  k:=p;
  while k>0 do
  begin
    for i:=1 to n do
      if (r[i,k]<>255) and (db[i]+k<=p)
                       and (df[i]+k<=p)
      then
        begin
          solutie(i,k); Close(g); Halt
        end;
      k:=k-1
  end;
  Writeln('NU')
end;

Begin
  citeste;
  rezolva;
  scrie
End.
```

Clasa a XII-a: **Bomboane**

Problemă propusă de prof. Dana Lica,
Liceul "I. L. Caragiale", Ploiești

Enunț:

La o fabrică de bomboane, sunt puse pe un stand n ($n \leq 100$) cutii de bomboane. Fiecare din cele n cutii conțin

câte m (m≤1000) bomboane dintr-un singur sortiment. Toate cele n sortimente sunt distincte. Un angajat mai distrat începe să se amuze și amestecă bomboanele din cutii. Spre a nu fi observată modificarea, el are grijă ca în fiecare cutie să rămână câte m bomboane.

Necazul apare când șeful său îi cere să-i aducă câte o bomboană din fiecare cutie, deci din fiecare sortiment. Fiind urmărit de către acesta, muncitorul nu va avea voie să extragă decât o bomboană din fiecare cutie.

Muncitorul acordă fiecărei extrageri un grad de risc. Astfel, la extragerea unei bomboane din sortimentul cu numărul i din cutia care conținea inițial sortimentul cu numărul j, gradul de risc va fi valoarea absolută a diferenței dintre i și j. Gradul de risc al tuturor celor n extrageri va fi egal cu suma riscurilor fiecăreia.

Să se determine ce sortiment de bomboană va trebui să extragă muncitorul din fiecare cutie pentru ca gradul de risc total să fie minim.

Date de intrare:

Pe prima linie a fișierului BOMBON.IN sunt scrise numerele n și m despărțite printr-un spațiu. Pe următoarele n linii se găsesc câte m numere despărțite printr-un spațiu. Cele m numere reprezintă sortimentele bomboanelor ce se regăsesc în fiecare cutie după amestecare, în ordine, începând cu cutia având numărul de ordine 1 până la cutia având numărul de ordine n.

Date de ieșire:

Fișierul de ieșire BOMBON.OUT conține două linii:

- pe prima linie se va scrie un număr natural reprezentând gradul total minim de risc;
- pe a doua linie sunt scrise n numere despărțite printr-un spațiu reprezentând numărul de ordine al cutiei din care va fi scoasă bomboana din fiecare sortiment, în ordine, începând cu sortimentul 1 până la sortimentul n.

În cazul în care există mai multe posibilități de extragere cu risc total minim, se va afișa una singură.

Exemplu:

BOMBON.IN	BOMBON.OUT
7 3	10
1 2 7	1 5 3 4 7 2 6
6 6 4	
4 7 3	
4 2 3	
2 1 3	
7 5 1	
5 5 6	

Timp maxim de executare/test: 6 secunde.

Rezolvare:

Abordarea rezolvării problemei printr-un algoritm poliminial ne conduce la modelarea ei cu ajutorul rețelelor de flux.

Se consideră un nod sursă și un nod destinație (artificial). Rețeaua va avea în afară de aceste două noduri încă 2n noduri, deci dublul numărului de sortimente distincte de bomboane. Vom uni pe rând fiecare dintre primele n noduri (reprezentând sortimentele distincte de bomboane, numerotate de la 1 la n), considerând că aceste arce au capacitatea 1 și costul 0. Vom uni apoi celelelalte n noduri (reprezentând cele n cutii, numerotându-le de la n+1 la 2n) cu nodul destinație, prin arce de capacitate 1 și cost 0. Apoi vom pune arce între nodul i (1≤i≤n) și nodul j (n+1≤j≤2n) dacă bomboana din sortimentul i se regăsește în cutia j-n după amestecare. Acest arc va avea capacitatea 1 și costul egal cu valoarea absolută dintre j-n-i (datorită numerotării cutiilor începând cu n+1). Rețeaua fiind modelată, problema revine la calculul fluxului de valoare maximă și de cost minim.

Algoritmul constă în mărirea continuă a fluxului cu valoarea dată de drumul de cost minim între nodul sursă și nodul destinație, (drum format doar din arce nesaturate).

Determinarea drumului minim se va face cu algoritmul lui *Roy-Floyd* întrucât el acceptă costuri negative pe arce.

Listing BOMBOANE.PAS

```
Program bomboane_amestecate;
const maxn=100;
type lin=array  [0..2*maxn+1] of Integer;
     plin=^lin;
     mat=array [0..2*maxn+1] of plin;
var cap,cost:mat;
    n,m,i,j,k,t,sursa,dest,v:Integer;
    d,tata:array [0..2*maxn+1] of Integer;
    actua:Boolean; f:Text;

  procedure aloca;
  begin
    for i:=0 to dest do
    begin
      New(cap[i]); New(cost[i]);
      for j:=0 to dest do
      begin
        cap[i]^[j]:=-1; cost[i]^[j]:=maxint
      end
    end;
    for i:=1 to n do
    begin
      cap[sursa]^[i]:=1;
      cost[sursa]^[i]:=0;
      cost[i]^[sursa]:=0
    end;
    for i:=n+1 to 2*n do
    begin
      cap[i]^[dest]:=1;
      cost[i]^[dest]:=0;
      cost[dest]^[i]:=0
    end
  end;
```

```
procedure load;
begin
  Assign(f,'BOMBON.IN');
  Reset(f);
  Readln(f,n,m);
  sursa:=0;
  dest:=2*n+1;
  aloca;
  for i:=1 to n do
  begin
    for j:=1 to m do
    begin
      Read(f,k);
      cap[k]^[i+n]:=1;
      cost[k]^[i+n]:=abs(i-k);
      cost[i+n]^[k]:=cost[k]^[i+n]
    end;
    Readln(f)
  end;
  Close(f)
end;

procedure drumul_minim;
begin
  for i:=sursa to dest do
  begin
    d[i]:=maxint;
    tata[i]:=sursa
  end;
  d[sursa]:=0;
  repeat
    actua:=false;
    for i:=sursa to dest do
      if d[i]<>maxint
      then
        for j:=sursa to dest do
          if cap[i]^[j]=1
          then
            if d[i]+cost[i]^[j]<d[j]
            then
              begin
                d[j]:=d[i]+cost[i]^[j];
                tata[j]:=i;
                actua:=true
              end
  until not actua
end;

procedure umple_sigur;
begin
  i:=dest;
  while i<>0 do
  begin
    j:=tata[i];
    cap[j]^[i]:=0;
    cap[i]^[j]:=1;
    i:=j
  end
end;

procedure umple;
begin
  drumul_minim;
  umple_sigur
end;

function gata:Boolean;
begin
  v:=0;
  for i:=n+1 to 2*n do
    if cap[i]^[dest]=0 then v:=v+1;
  gata:=v=n
end;

procedure scrie;
begin
  t:=0;
  for i:=1 to n do
  begin
    k:=0;
    for j:=n+1 to 2*n do
      if (cap[i]^[j]=0) and cost[i]^[j]<>maxint)
      then k:=j;
    t:=t+cost[i]^[k]
  end;
  Assign(f,'BOMBON.OUT'); Rewrite(f);
  Writeln(f,t);
  for i:=1 to n do
  begin
    k:=0;
    for j:=n+1 to 2*n do
      if (cap[i]^[j]=0) and cost[i]^[j]<>maxint)
      then k:=j;
    Write(f,k-n,' ')
  end;
  Close(f)
end;

Begin { programul principal }
  load;
  repeat umple until gata;
  scrie
End.
```

Programe Sursă:
Puteți descărca listingurile prin Internet de la adresa http://www.ginfo.ro.

Paginile soluții au fost realiza de **Clara Ionescu**,
e-mail cionescu@ginfo.ro.

PROBLEME propuse pentru REZOLVARE

Probleme, probleme... Unele preluate de la concursuri ACM, (prin bunăvoința studentului Valer Crișan și împreună cu altele, propuse membrilor lotului național de informatică cu ocazia taberei de antrenament de la Liceul de Informatică din Cluj.

P039801: În fabrica de mobilă

Într-o fabrică se confecționează din plăci dreptunghiulare, piese de mobilier, de asemenea dreptunghiulare, de diferite dimensiuni.

Mașina care decupează aceste piese impune următoarele restricții:

— o tăietură realizată într-o placă trebuie să o separe pe aceasta în două plăci (trebuie să treacă dintr-o parte în alta, drept, fără a roti cuțitul);

— prin tăiere se pierde din placă o bucată de dimensiune Δ.

Despre fiecare piesă solicitată se știe dacă se pot roti (sau nu) cu 90° în scopul confecționării din placa existentă. Rotirea necesită repoziționări ale cuțitului mașinii.

Să se determine modalitatea de dispunere a pieselor solicitate pe plăcile existente, astfel încât materialul pierdut cât și numărul de repoziționări ale cuțitului să fie minime.

Date de intrare:

Pe prima linie a fișierului de intrare `PLACI.IN` sunt date dimensiunile plăcilor existente (lățime și lungime, numere naturale ≥1 și ≤1000), numărul lor fiind infinit și toate plăcile fiind de aceeași dimensiune.

Pe linia a doua este scrisă valoarea lui Δ (număr natural, 1≤Δ≤5).

Pe următoarele linii sunt scrise date referitoare la tipurile de piese solicitate, despărțite printr-un spațiu:

`nr latime lungime caracter`

unde:

— `nr` este un număr natural care reprezintă numărul de piese solicitate de același tip (1≤nr≤200);

— `latime` și `lungime` precizează dimensiunile piesei solicitate (numere naturale, 1≤latime, lungime≤1000);

— `caracter` este fie `'r'` fie `'n'`; prin ele se precizează dacă o piesă poate fi (sau nu) rotită cu 90° la aranjarea pe placă.

Date de ieșire:

Rezultatele se vor scrie în fișierul de ieșire `PLACI.OUT` astfel:

`pierdere` *// reprezintă suprafața totală a materialului pierdut din plăcile folosite (total Δ și material din care nu se poate confecționa nici o piesă solicitată);*

`nr_rotiri` *// numărul de repoziționări ale cuțitului;*

`placa x` *// cuvântul* `'placa'`*, un spațiu, apoi* `x` *= numărul de ordine al plăcii;*

`piesa st sus caracter` *// unde* `piesa` *este numărul de ordine al piesei,* `st` *și* `sus` *sunt coordonatele poziției piesei pe placă (originea =*`0,0` *se consideră a fi în colțul din stânga sus al plăcii);*

// `caracter` *este* `'r'` *sau* `'n'` *în funcție de posibilitățile de rotire.*

P039802: Cămile

O caravană cu `n` (n≤30) cămile înaintează în deșert în șir indian. După prima zi de mers apare un conflict între fiecare cămilă și cea din fața sa (între a doua și prima, între a treia și a doua etc).

Conducătorul caravanei va trebui ca a doua zi, datorită conflictelor, să schimbe ordinea cămilelor în caravană. Să se găsească numărul de posibilități pe care le are conducătorul caravanei.

Date de intrare:

Din fișierul `CAMILE.IN` se va citi numărul natural `n` al cămilelor.

Date de ieșire:

Rezultatul se va scrie în fișierul `CAMILE.OUT`.

Exemplu:

```
CAMILE.IN      CAMILE.OUT
    3              3
```

P039803: HDTV (High Definition TeleVision)

La televiziunea HDTV se utilizează mai multe modalități de comprimare a imaginilor în scopul eliminării informațiilor redundante, astfel reducând cantitatea de informații care urmează să fie transmisă. Să se simuleze decomprimarea în scopul reconstruirii imaginilor de televiziune. Pentru simplificare, imaginile sunt reprezentate prin tablouri dreptunghiulare de pixeli având originea (0,0) în colțul din stânga jos. Fiecare pixel poate avea ca valoare una din cele 26 de litere ale alfabetului englez de la 'A' la 'Z'.

Date de intrare:

Sistemul de comenzi preia o literă mică urmată de lista parametrilor care reprezintă comanda. Anumiți parametri definesc blocuri dreptunghiulare din imagine. În listele de comenzi de mai jos, blocul (x,y,w,h) reprezintă un bloc dreptunghiular având colțul din stânga jos în punctul de coordonate (x,y) și colțul din dreapta sus în punctul de coordonate (x+w-1,y+h-1). Aceste comenzi și parametrii lor sunt:

```
i          x y
```

Efect:

Se *inițializează* o imagine nouă având lățimea de x pixeli și înălțimea de y pixeli. Imaginea conține cel mult 10000 pixeli care sunt inițializați cu litera 'A'. Ultima comandă în fișierul de intrare este o comandă similară care indică o imagine având dimensiunea 0.

```
f          x y w h c
```

Efect:

Se *umple* blocul (x,y,w,h) cu caracterul c, unde c este o majusculă.

```
c          x y w h a b
```

Efect:

Se *copiază* un bloc (x,y,w,h) în blocul (a,b,w,h). Aceste blocuri se pot suprapune.

```
r          x y w h data
```

Efect:

Se *citesc* valorile elementelor cu care se va încărca blocul (x,y,w,h) din lista de date data. Blocul este umplut linie după linie începând cu pixelul (x,y). Datele sunt wxh majuscule.

```
d          x y w h data
```

Efect:

Se *decomprimă* datele dintr-un bloc (x,y,w,h) în mod similar comenzii r. Datele sunt memorate într-un format codificat, alcătuit din perechi de *număr-literă*, unde *număr* precizează numărul de apariții al literei *literă*.

Exemplu:

3A1B10C corespunde șirului decomprimat AAABCCCCCCCCCC.

Observații:

— Comenzile și parametrii lor pot conține spații și tabulatoare și pot fi continuate pe mai multe linii.
— Un număr nu va fi rupt în cifre niciodată.
— Parametrii sunt numere pozitive și sunt de așa natură încât orice bloc specificat se află în interiorul imaginii.
— În cazul comenzilor r și d, conținutul parametrului data respectă dimensiunile blocului.

Din fișierul de intrare TV.IN se va citi o listă de comenzi.

Date de ieșire:

De fiecare dată când se întâlnește o comandă i, unitatea de control va calcula un număr de verificare pentru imaginea existentă, apoi se va inițializa o imagine nouă. Numărul de verificare este *suma modulo 256 a codurilor ASCII ale caracterelor corespunzătoare pixelilor din imagine, luate linie după linie* în forma:

```
nr_de_verificare = 127
```

cu un singur spațiu în fața și după semnul '='.

Nici un număr de verificare nu se va calcula și scrie pentru prima comandă i, deoarece nu există nici o imagine în acest moment.

Următoarele diagrame prezintă forma imaginilor corespunzătoare conținutului fișierelor TV.IN și TV.OUT.

```
                AAAAUGHAAA      LOVE
                BBBBLEAAAA      LONE
                BBBBLLLAAA      LATE
  AA            AAAABLLAAA      HATE
```

Exemplu:

TV.IN

```
    i2 1i 10 4f 0 1 6 2 Bd4 0 3
    4 1B6L1E1A
    1U1G1H i 4 5 r 0 4 4 1 L  OVE c 0 4 4
    1 0 3 f 2 3 1 1 N
    c 0 3 4 1 0 2 f 1 2 1 1 A c 0 2 4
    1 0 1 r 2 1 1 1 T
    c 0 1 4 1 0 0 f 0 0 1 1 Hi 0 0
```

TV.OUT

```
    nr_de_verificare = 130
    nr_de_verificare = 152
    nr_de_verificare = 204
```

P039804: Inteligență artificială simbolică

Trebuie să scrii un program simplu de inteligență artificială simbolică. Acestuia i se vor furniza anumite informații și i se vor pune întrebări. Programul de inteligență artificială trebuie să răspundă la întrebări. Din fericire aceste informații și întrebări se referă numai la anumiți indivizi care sunt grupați (pentru grupare se folosesc substantive colective) care efectuează acțiuni (pentru gruparea acțiunilor se folosesc verbe).

Gramatica folosită este foarte simplă: se folosesc două tipuri de propoziții și un singur tip de întrebare. Toate

substantivele la plural sunt marcate printr-un caracter 's'.

Date de intrare:
Fiecare șir din datele de intrare conține fie o singură propoziție, fie un spațiu. Sfârșitul fișierului este marcat de o linie a cărui conținut este caracterul '#'. Fiecare propoziție are una din următoarele forme:

— 'as v bs.', unde a și b sunt substantive și v este un verb. Aceasta înseamnă că tot ce este de tip a realizează acțiunea v asupra a tot ce este de tip b.
— 'a is a b.', unde a și b sunt substantive. Aceasta înseamnă că individul a este de tip b.
— întrebarea are forma: 'does a v b?', unde a și b sunt substantive, iar v este verb. Se pune întrebarea: individul a face acțiunea v asupra individului b?

Lungimea maximă a șirurilor este de 80 litere. Cuvintele conțin cel mult 40 de litere și pot fi formate din majuscule și litere mici, deși în timpul căutării litera mică nu se deosebește de majusculă. Pot exista mai multe spații între cuvinte, dar niciodată între cuvinte și semnele de punctuație de la sfârșitul propozițiilor. Numărul maxim de informații dintr-un fișier de intrare este 100.

Date de ieșire:
Pentru fiecare întrebare din fișierul de intrare se va afișa o singură linie, având conținutul: 'YES.' în cazul în care din informațiile anterioare se poate decide că răspunsul la întrebare este *adevărat* și 'NO.' în caz contrar.

Exemplu:

AI.IN	AI.OUT
Dogs chase cats.	YES.
Sheeps ignore glasss.	NO.
Fido is a dog.	
Fluffles is a cat.	
Does Fido chase Fluffles?	
Does Fluffles chase Fido?	

P039805: Potrivire de cuvinte

Ca agent al serviciilor secrete, ai sub supraveghere traficul corespondenței pe poșta electronică, căutând coduri care ar putea să indice acțiuni subversive. Totuși, anumite elemente subversive ale societății au început să amestece literele din unele cuvinte din mesajele lor cu scopul de a le face de neînțeles. Tu, familiarizat cu asemenea trucuri, te decizi să scrii un program care să caute propoziții având cuvinte ale căror litere au fost amestecate.

Date de intrare:
Datele de intrare din fișierul AI.IN conțin perechi formate din liste de cuvinte și propoziții. Lista de cuvinte conține cuvintele care trebuie căutate în propoziția următoare. Toate literele din datele de intrare sunt litere mici. Cuvintele pot fi formate din cel mult 20 de litere și sunt separate printr-un spațiu. Lungimea maximă a fiecărei linii este de 80 de caractere. Fiecare listă este scrisă pe o singură linie și conține cel mult 20 de cuvinte. Propozițiile se pot continua pe mai multe linii și se termină cu caracterul '.'.

Fișierul se termină cu o listă care conține caracterul '#'.

Date de ieșire:
Pentru fiecare pereche de listă de cuvinte și propoziție, să se găsească cuvintele din listă care apar în propoziții (având caracterele amestecate sau nu). În fișiereul AI.OUT se vor afișa aceste cuvinte în ordinea în care apar în listă, separate printr-un spațiu. Dacă nici un cuvânt din listă nu se

OAC...

Problemă propusă de **Cristian Dimache**, *elev la Colegiul de Informatică "Tudor Vianu", București.*

Într-o mlaștină, o broască zărește în depărtare o libelulă, situată pe un nufăr. Broasca se poate deplasa din nufăr în nufăr până pe nufărul pe care se află libelula, pentru a-și astâmpăra foamea. Broasca poate sări din nufăr în nufăr asemeni calului la șah.

Generați, pentru o poziție inițială, una dintre posibilitățile broaștei de a se sătura.

Date de intrare:
Fișierul 'OAC.IN', cuprinzând:
- pe prima linie latura n pentru matricea pătratică a mlaștinii;
- pe următoarele n linii, câte n caractere separate prin spațiu, simbolizând: 'B' - broască, 'L' - libelulă, 'N' - nufăr, 'M' - mlaștină;

Date de ieșire:
În fișierul 'OAC.OUT' se va scrie:
- "imposibil", dacă broasca nu poate ajunge la destinație în condițiile date;
- matricea pătratică a drumului parcurs de broască. Se vor marca cu 'i' pozițiile în care broasca aterizează, inclusiv cea finală (i = numărul de ordine al săriturii).

Reguli:
drumul parcurs este continuu, fără abateri, reîntoarceri sau sărituri în afara mlaștinii;
1<n<50;
0<numărul de sărituri<n2-1.

Exemplu:

OAC.IN	OAC.OUT
6	B N N N M N
B N N N M N	M N 1 M N N
M N N M N N	2 M N M N M
N M N M N M	N N 3 N N M
N N N N N M	M N N N M N
M N N N M N	M 4 N N M N
M L N N M N	

potrivește cu vreun cuvânt din propoziție, se va afișa o linie vidă.

Exemplu:

```
CUVINTE.IN
      bomb
      we twan ot ombb mericaa.
      guns mihes missiles
      aameric ssell snug
      dan iimsssle ot sit neeemis.
      cat sat mat
      cat sat tam mat.
      #
CUVINTE.OUT
      bomb
      guns missiles
      cat sat mat
```

P039806: Sfârșitul lumii

Unui profet i s-a spus că sfârșitul lumii poate fi cândva între 01994 și 99999 inclusiv. I s-au dat diferite indicații privind data posibilă, prezentate sub forma:

"Cifra zecilor este de trei ori mai mare decât cifra sutelor, plus șase".

Profetul a interpretat această indicație sub forma: `'d=3c+6;'`

(e este cifra zecilor de mii, d este cifra miilor, c este cifra sutelor, b este cifra zecilor, a este cifra unităților numărului care reprezintă un an).

Să se scrie un program care citește un set de indicații care pot avea una din formele:

<*v*> = *propoziție* +/- *propoziție*, sau
<*v*> = *propoziție*.
unde:
propoziție poate fi:
k<*v*> sau
<*v*> sau
k
unde:
<*v*> este o variabilă (a, b, c, d, sau e);
k este o constantă având valoarea între 0 și 9.

Date de intrare:

Fiecare linie din fișierul de intrare `LUMEA.IN` conține un set de indicații diferite. Sunt cel mult 5 indicații în fiecare set. Să se afișeze, pentru fiecare set, prima dată după 1993 care poate fi un an care să reprezinte sfârșitul lumii (nu contrazice nici una din indicații), sau mesajul `'Suntem salvati!'`, dacă rezultă că nici o dată până la 100000 nu ar fi data probabilă a sfârșitului lumii (de asemenea, această concluzie nu contrazice nici una din indicațiile setului).

Toate calculele sunt realizate modulo 10. Fișierul de intrare se termină printr-o linie care conține caracterul `'#'`.

Exemplu:

```
LUMEA.IN
      a=2;b=4;e=9+a;
      d=3;
      c=e+e;d=4;c=d+e;
      e=e+1;
      c=2d+4;e=4-2c;a=4d+5;d=3;b=a-e;
      #
LUMEA.OUT
      10042
      03000
      44800
      Suntem salvati!
      43037
```

P039807: Prelucrare de texte

Să se scrie un procesor de texte care va prelucra șiruri de caractere (string-uri). Va citi o singură linie a textului, urmată de o linie de comenzi care va modifica șirul. Programul trebuie să producă la ieșire o linie a textului prelucrat.

Comenzile sunt următoarele:

- `0` — mută cursorul la începutul liniei;
- `$` — mută cursorul în poziția următoare ultimului caracter (în dreptul caracterului de sfârșit de linie);
- `x` — șterge caracterul din dreptul cursorului dacă nu este la sfârșitul liniei;
- `s` — înlocuiește caracterul din dreptul cursorului cu caracterul din dreapta acestuia, cât timp cursorul nu este în fața caracterului sfârșit de linie sau în dreptul acestuia;
- `i x` — inserează caracterul x pe poziția cursorului, și mută cursorul cu un caracter;
- `u` — schimbă litera mică în majusculă și mută cursorul cu un caracter;
- `+` — mută cursorul cu un caracter la dreapta;
- `-` — mută cursorul cu un caracter la stânga.

Cursorul poate fi poziționat în dreptul primului caracter, sau în dreptul oricăruia din fața caracterului de sfârșit de linie.

Comenzile din fișierul de intrare `TEXTE.IN` sunt scrise pe o linie, după care, pe o linie nouă urmează textul care nu conține spații. Linia în care s-a scris textul nu conține (în interiorul textului) caracterul de sfârșit de linie. Cursorul pornește de la primul caracter al fiecărui text. Fișierul se termină cu o linie text care conține un singur caracter: `'#'`.

Exemplu:

```
TEXTE.IN
      Hello, I am a frog.
      $-----xxxxipieirisioin
      Needle nardle noo.
      +++xizuu+++xxips
      #
TEXTE.OUT
      Hello, I am a person.
      NeezLE napel noo.
```

P039808: Trupa de comando

Se consideră o hartă pe care sunt desenate n țări. Suprafața fiecărei țări este colorată într-o singură culoare.

O trupă de comando pornește dintr-o țară dată x și trebuie să ajungă într-o altă țară dată y. Pe parcursul traversării oricărei țări, membrii trupei trebuie să se îmbrace în echipament de camuflaj având culoarea de pe hartă a țării traversate. Determinați drumul trupei de comando astfel încât ei să aibe nevoie de un număr minim de echipamente.

Date de intrare:

Fișierul de intrare INPUT.TXT are următoarea structură:

— pe prima linie se află un număr întreg n ($2 \le n \le 100$), reprezentând numărul de țări de pe hartă;

— fiecare dintre următoarele n linii conține un string care are structura:

$nume_i$ $culoare_i$ $vecin_1$ $vecin_2$ ... $vecin_k$

unde:

— $nume_i$ reprezintă numele unei țări (string);

— $culoare_i$ reprezintă culoarea (string) atașată țării având numele $nume_i$;

— $vecin_1$ $vecin_2$ ... $vecin_k$ ($0 \le k \le n$) reprezintă numele țărilor vecine cu țara $nume_i$;

— șirurile de caractere corespunzătoare numelor, respectiv culorii sunt separate prin câte un spațiu;

— pe următoarea linie sunt date două nume (stringuri): țara din care pornește expediția și țara în care trebuie să ajungă.

Se presupune că datele sunt corecte.

Date de ieșire:

Fișierul de ieșire OUTPUT.TXT are următoarea structură:

— pe prima linie se va scrie un număr întreg m ($1 \le m \le n$) reprezentând numărul de echipamente necesare trupei pentru a realiza expediția respectând cerințele problemei;

— pe următoarele m linii se va descrie drumul trupei, specificând pe fiecare linie numele unei țări prin care va trece trupa.

În cazul în care există mai multe soluții, se va specifica cea care trece printr-un număr minim de țări.

Exemplu:

INPUT.TXT:

```
6
Romania rosu Bulgaria Ungaria Yugoslavia
Bulgaria verde Romania Yugoslavia
Ungaria galben Romania Yugoslavia Croatia
Yugoslavia alb Romania Ungaria Croatia
Croatia rosu Ungaria Yugoslavia Italia
Italia galben Croatia
Romania Italia
```

OUPUT.TXT:

```
2
Romania
Ungaria
Croatia
Italia
```

Doors 95

Problemă propusă de Alexandru Roșiu, elev la Liceul Teoretic "Traian Lalescu", Reșița.

Nu cu mult timp în urmă, într-un oraș american, o echipă de informaticieni a pus la punct un nou sistem de operare. Spre deosebire de cele create până atunci, acest SO permitea rularea simultană a mai multor programe. Cu câteva zile înainte de lansarea oficială, un spion al unei mari firme de software a penetrat sistemul de securitate al serverului paginii web deținute de grupul de talentați informaticieni și a sustras algoritmul de funcționare al SO-ului. Directorul firmei, dl. B.G., a fost foarte mulțumit de treaba făcută de spion și l-a răsplătit "gras".

Firma a finalizat proiectul înaintea adevăraților autori iar produsul lor, Doors95 s-a bucurat de mult succes. Ca orice alt produs al respectivei firme, Doors95 era plin de bug-uri și a fost curând uitat, deși era foarte scump. Spionul (tu), care avea ceva habar despre programare, a reținut o copie a algoritmului și își propune să creeze un sistem de operare bazat pe acesta, însă lipsit de bug-uri.

Algoritmul:

Datorită limitărilor hardware, simultan pot rula maxim N procese, cotate cu câte un număr Px, reprezentând prioritatea de execuție. Fiecare proces rulează o durată de timp individuală de Sx secunde.

Procesele sunt citite din fișierul PROCESE.IN, cu următoarea structură:

N
NumeProces1/s1/p1
NumeProces2/s2/p2
...
NumeProcesN/sN/pN

unde:

N este numărul de procese ($1 <= N <= 100$);

NumeProcesX este un șir de lungime maximă 10 caractere;

Sx este un număr întreg între 1 și 60;

Px este un număr întreg între 1 și 9 (1 = prioritate maximă);

Spionul și-a dat totuși seama că nu este în stare să finalizeze SO-ul, așa că s-a rezumat la determinarea timpului necesar terminării fiecărui proces, pe care programul îl scrie în fișierul PROCESE.OUT.

Exemplu:

PROCESE.IN	PROCESE.OUT
5	Proces1 140 s
Proces1/5/7	Proces2 280 s
Proces2/45/5	Proces3 40 s
Proces3/10/1	Proces4 228 s
Proces4/50/2	Proces5 172 s
Proces5/30/3	

FRACTALI și sisteme de numerație

de Stelian Ciurea

Figuri ciudate și aparent paradoxale, fractalii exercită o adevărată fascinație pentru matematicieni și informaticieni. Dincolo de spectaculosul imediat al reprezentărilor, fractalii și-au găsit aplicații nu doar în programele de grafică, ci și în domenii cum ar fi compresia datelor sau criptografie. Acest articol vă propune o incursiune în lumea fractalilor pornind de la niște noțiuni de aritmetică.

Sisteme de numerație

Se știe că ansamblul de reguli care ne permite să reprezentăm numere, cu ajutorul unor simboluri, constituie un **sistem de numerație.** Având drept criteriu de clasificare semnificația și valoarea ce se atribuie fiecărei poziții în cadrul reprezentării, există sisteme poziționale și sisteme apoziționale.

Notația pozițională, folosind *baza b*, este definită prin formula:

$(...a_3a_2a_1a_0.\ a_{-1}a_{-2}...)_b = ...+a_3b^3 + a_2b^2 + a_1b + a_0 + a_{-1}b^{-1} + a_{-2}b^{-2}+...$ (1)

Sistemul de numerație "uman" este cel *zecimal* în care *b=10 iar a_i sunt simbolurile 0,1,...,9* denumite și cifre zecimale. Faptul că la acest sistem $b\in N$, $b>1$ iar simbolurile ai sunt numere întregi care îndeplinesc condiția $0\le a_i<b$ ne conduce și la alte sisteme asemănătoare.

Pentru informatică prezintă interes sistemul *binar* cu *b=2, $a_i\in\{0,1\}$.* Din păcate, avantajul existenței doar a celor două simboluri, care permite asocierea acestora cu cele două stări ale circuitelor ce compun un calculator, ascunde și unul din dezavantajele sistemului: numărul mare de cifre necesare pentru a reprezenta valori care, însă în sistemul zecimal necesită un număr mic de simboluri: pentru un număr întreg exprimat pe n cifre zecimale avem nevoie de $[n*\log_2 10]+1 = [3.322*n]+1$. De exemplu, pentru numere zecimale de 3 cifre, ne sunt necesare numere binare cu 10 simboluri, iar pentru numere zecimale de ordinul miliardelor avem nevoie în binar de 33 de cifre.

Pentru a ușura exprimările numerelor binare avem la dispoziție sistemele la care baza este o putere a lui 2, conversiile între sistemul binar și acestea putându-se face fără a fi necesare adunări, împărțiri sau alte operații. Uzuale sunt sistemele *octal* cu *b=8, $a_i\in\{0,1,...,7\}$* și *hexazecimal* cu *b=16, $a_i\in\{0,1,...,9,A,B,...,F\}$.*

Mulțimi S

Pentru un sistem de numerație cu baza b și simbolurile a_k, mulțimea S notată S_b, este definită ca mulțimea formată din toate numerele care pot fi scrise sub forma

$$\sum_{k=1}^{\infty} a_k b^{-k} \qquad (2)$$

O astfel de mulțime este importantă deoarece ne permite să studiem dacă un sistem de numerație permite reprezentarea unei valori oricât de mici. În cazul bazelor de numerație naturale, calculul plajei de valori pentru o astfel de mulțime este imediată: de exemplu, pentru sistemul zecimal, valoarea cea mai mică a mulțimii S - $\inf(S_{10})$ este 0.00000... = 0 iar valoarea cea mai mare - sup(S10) este 0.9999... = 1. Această plajă de valori este valabilă pentru orice bază naturală.

Condițiile $b\in N$, $b>1$ și $0 \le a_i < b$, nu trebuie privite însă ca obligatorii; în general, se poate considera ca bază orice număr nenul și diferit de 1, iar simbolurile se pot alege dintr-o mulțime de valori specificate. Apar astfel o serie de sisteme de numerație interesante. Studiile făcute au urmărit găsirea unor baze care să permită reprezentări cât mai simple, efectuarea rapidă a calculelor, micșorarea numărului de cifre necesar reprezentării valorilor etc.

Sisteme cu baze negative

Tratate pentru prima oară la sfârșitul secolului trecut (1885), cu ele s-a urmărit înlăturarea unui neajuns aparent minor: reprezentarea numerelor negative cu ajutorul semnului -(minus). Astfel, folosind *baza -2* și *simbolurile 0 și 1* se obțin următoarele reprezentări, pentru care, evident, nu mai este nevoie de minus:

$10 = 11110_{-2}$; $9 = 11001_{-2}$; $8 = 11000_{-2}$; $1 = 1_{-2}$;
$-10 = 1010_{-2}$; $-9 = 1011_{-2}$; $-8 = 1000_{-2}$; $-1 = 11_{-2}$.

Au fost studiate, pentru astfel de sisteme, reguli pentru efectuarea operațiilor și conversiilor, teste de divizibilitate, extrageri de rădăcini etc.

Este interesant de subliniat că și în cazul acestor baze, diferența între valoarea maximă din mulțimea S și cea minimă este tot 1. Mai exact, pentru un sistem cu baza -b, (b pozitiv):

$$\sup(S_{-b}) = 1/(b+1),\ \inf(S_{-b}) = -b/(b+1) \quad (3)$$

deci $\sup(S_{-b})-\inf(S_{-b}) = = 1/(b+1) -(-b/(b+1)) = 1.$ (4)

Demonstrarea egalităților (3) constituie o utilă aplicație a limitelor de șiruri și de aceea o propun ca exercițiu celor care au studiat materia clasei a XI-a.

Sistemul ternar balansat

De o atenție deosebită s-a bucurat sistemul *ternar balansat*. La acesta *b=3* iar simbolurile folosite, denumite *triți* (prin analogie cu biți) sunt *-1, 0, 1*. Fără a intra în amănunte, sunt demne de semnalat câteva proprietăți ale acestui sistem:

- negativul unui număr se obține prin schimbarea simbolurilor 1 cu -1 și reciproc, -1 cu 1;
- două numere reprezentate în acest sistem se compară după reguli similare cu cele din sistemul zecimal;
- semnul unui număr astfel reprezentat este dat de simbolul cel mai semnificativ diferit de 0;
- operația de rotunjire are efect identic cu cea de trunchiere;
- numărul de cifre necesar pentru a reprezenta în ternar balansat o valoare zecimală oarecare este aprox. 63% din cel folosit pentru a reprezenta valoarea respectivă în sistemul binar.

Ar mai fi de adăugat că realizarea de circuite care să lucreze cu trei stări nu presupune o complexitate mult mai mare decât realizarea celor cu două stări și că în anii '50 sovieticii au realizat un calculatorul denumit SETUN și bazat pe sistemul ternar balansat...

Sisteme cu baze complexe

Anii '50 au consemnat o serie de studii privind sistemele de numerație cu baze complexe. Pentru scopul inițial al articolului prezintă interes cel cu baza *i-1*. Acesta folosește, la fel ca sistemul binar, doar simbolurile *0* și *1*. El a fost prezentat pentru prima dată de W. Penney într-o lucrare publicată în anul 1965. Un număr scris în această bază poate fi convertit în zecimal cu formula:

$$(...a_4a_3a_2a_1a_0a_{-1}...)_{i-1} =$$
$$= ... -4a_4 + (2+2i)a_3 -2ia_2 + (i-1)a_1 + a0 -1/2(i+1)a_{-1} + ... \quad (5)$$

S-a pus problema dacă orice număr complex poate fi reprezentat sub această formă. Astfel s-a studiat mulțimea Si-1, care pentru această bază conține orice element de forma:

$$(0.a_{-1}a_{-2}a_{-3}...)_{i-1} = \sum_{k=1}^{\infty} a_k(i-1)^{-k} \quad , \quad (6)$$

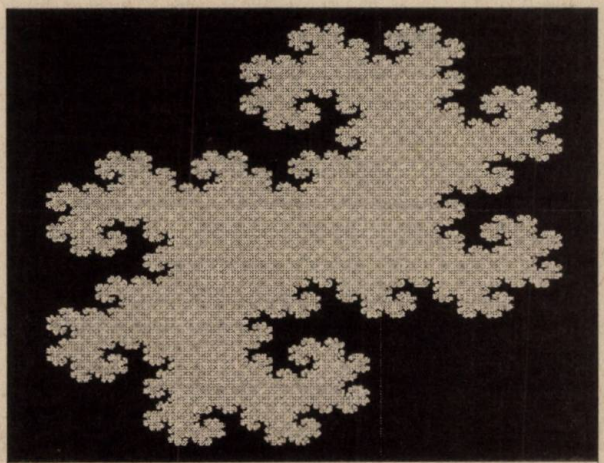
unde a_{-1}, a_{-2}, ..., a_k,.. sunt 0 sau 1.

Apelându-se la metode ce țin de analiză în spații complexe, s-a demonstrat că sistemul permite reprezentarea oricărui număr complex, dar totodată, s-a constatat că punctele ce aparțin mulțimii S_{i-1} constituie, în planul complex, o figură ce are proprietatea că poate fi descompusă în părți din ce în ce mai mici, dar care sunt similare cu figura inițială. Benoit B. Mandelbrot a definit o astfel de figură ca fiind un *fractal*.

Fractali

Obiectele studiate în geometria de liceu sunt obiecte a căror mărime se poate preciza exact. De exemplu, se poate da valoarea exactă pentru lungimea unei drepte între două puncte, suprafața unui pătrat sau volumul unui cub. Astfel, s-a postulat că diversele obiecte geometrice au ca dimensiuni topologice valori întregi: dreptele și curbele 1, suprafețele plane 2, spațiul 3. Aceste dimensiuni au fost extinse pentru a include o a patra dimensiune teoretică și au fost imaginate dimensiuni întregi mai mari.

Însă, pe la începutul secolului nostru, matematicienii au reușit să construiască figuri care nu puteau fi caracterizate prin metodele de până atunci: de exemplu, nu se putea preciza lungimea unei astfel de curbe între două puncte. Mai mult, făcându-se detalieri succesive de-a lungul fiecărui punct al curbei, lungimea creștea. Chiar dacă o astfel de comportare pare ciudată, ea este mult mai apropiată de ceea ce ne înconjoară: conturul unui munte conține din ce în ce mai multe detalii pe măsură ce ne apropiem de el. În 1919, un matematician devenit celebru - Hausdorff, a introdus noțiunea de dimensiune fractală (cunoscută ca "dimensiune Hausdorff") care a permis caracterizarea unor astfel de figuri. Intuitiv, această dimensiune măsoară mărimea numărului de mulțimi de diametre mai mici decât



"Dragonul cu două capete" (n=18, nfig=1)

ε, necesare pentru a acoperi respectiva figură când ε→0. Această noțiune este folosită pentru a da una din definițiile acceptate pentru fractali:

Un fractal este o figură la care dimensiunea Hausdorff depășește dimensiunea topologică.

Pentru exemplificare, fără a intra în amănunte, voi aminti o curbă fractală celebră denumită "curba dragonului" care se obține plecând de la un segment de dreaptă care este înlocuit cu cele două catete cu care el ar forma un triunghi dreptunghic isoscel; în pasul următor, cele două catete sunt la rândul lor înlocuite după același principiu ș.a.m.d:

• dimensiunea topologică este 1 (curba este formată din segmente de dreaptă);

• dimensiunea Hausdorff (notată D) se obține din formula:

$n \cdot r^D = 1$, cu n=numărul de contracții=2 și r=rația unei contracții=$1/\sqrt{2}$,

de unde $D = \lg 2/\lg \sqrt{2} = 2\sqrt{2}$

Programul pentru reprezentarea mulțimii S_{i-1}

Programul, scris în Turbo Pascal, va face această reprezentare desenând în pozițiile corespunzătoare din planul complex, punctele ce formează mulțimea Si-1, conform formulei (6), în care se înlocuiește secvența infinită de cifre binare cu o secvență finită de n cifre; aceste numere sunt convertite din baza i-1 în baza 10 conform formulei:

$(0.a_{-1}a_{-2}a_{-3}...a_n)_{i-1} = a_{-1}(i-1)_{-1} + a\text{-}2(i-2)_{-2} + ... + a_{-n}(i-1)^{-n}$ (6')

Pentru calculul puterilor negative ale numărului i-1 se folosesc formulele:

$(i-1)^{-4k}=(-1/4)^k$

$(i-1)^{-(4k+1)}=(1/2)(-1/4)^k\,(i+1)$

$(i-1)^{-(4k+2)}=(1/2)(-1/4)^k\,i$

$(i-1)^{-(4k+3)}=(1/4)(-1/4)^k\,(i-1)$ (7)

Implementarea acestor formule se face prin funcțiile putere și pow.

Pentru a genera toate combinațiile de 0 și 1 pentru cele n cifre, se folosește reprezentarea internă a unei variabile de tip `longint` notată i, care ia din 1 în 1 toate valorile dintre 0 și $(11...1)_2 = 2^n-1$ (având în vedere că sunt n cifre de 1). Funcția **citeste_bit** determină dacă în cadrul reprezentării interne a valorii i, bitul de pe poziția dată ca parametru este în 0 sau 1.

Desenarea efectivă se face prin `putpixel` în care coordonata pe orizontală corespunde părții reale a numărului, iar coordonata pe verticală - părții imaginare. Trecerea din coordonatele planului complex în cele ale ecranului se face prin funcțiile **xprim** pentru axa reală și **yprim** pentru cea imaginară. Implementarea acestora presupune cunoașterea domeniului de valori pe cele două axe. Acestea sunt:

• pentru axa reală minimul, notat `remin`=-13/15 și maximul, `remax`=7/15;

• pentru axa imaginară minimul, `immin`=-14/15 și maximul, `immax`=11/15.

(Aceste valori se obțin plecând de la formulele (6) și (7) în care se reține doar termenii reali sau pur imaginari pozitivi pentru maxim și negativi pentru minim; termenii reținuți sunt în progresie geometrică așa că trecerea la limită se poate face cu cunoștințele din clasa a XI-a.)

Acoperitor, programul lucrează cu domeniul [-1,1] pentru ambele axe.

Evidențierea caracterului fractal al figurii se poate face în două moduri:

• desenând figura descompusă într-un număr de figuri mai mici, egale între ele și având aceeași formă cu "figura mare"; numărul acestora este egal cu puterea la care apare 2 în descompunerea în factori primi a valorii `nfig` citită de la tastatură. Dacă `nfig` este chiar o putere a lui 2, figurile mici sunt desenate într-o singură culoare, altfel ele sunt colorate cu combinații de culori, și pentru anumite valori ale lui `nfig` nu mai apare o diferențiere netă între ele. În program, alegerea culorii pentru figurile mici se face în funcție de restul împărțirii lui `i` la `nfig`;

• detaliind figura prin micșorarea modulului pentru `remax`, `remin`, `immax` și `immin` și folosind o valoare mare pentru `nfig`, se obțin imagini care reproduc figura inițială.

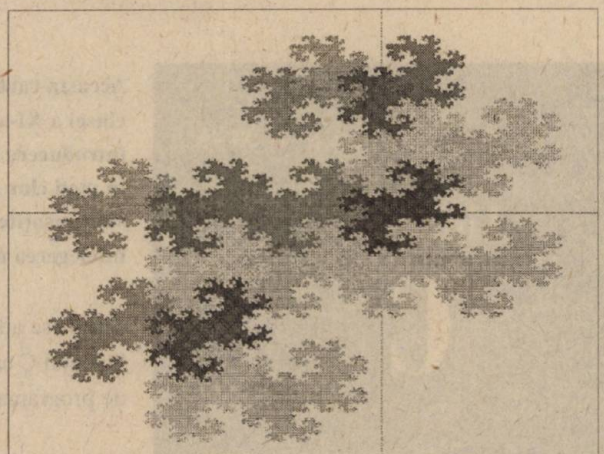
Benoit Mandelbrot a dat acestei figuri numele de "dragonul cu două capete" observând că imaginea mulțimii S_{i-1} se poate obține prin alipirea a două curbe ale dragonului la nivelul abdomenului.

Listing MULTIME.PAS

Programul este următorul:

```
program multimea_S;  {multimea S in baza i-1}
{$N+}
uses graph, crt;
const NrCfMax=32;
type complex=record re, im: Real end;
var gd, gm: Integer;
    remax, immax,remin,immin: Real;
    cul, nfig,n: Word;
    i, j: Longint;
    sum, z: complex;
function xprim(x:real):_Integer;
  begin
    xprim:=round((getmaxx-10)*(x-remin)/(remax-
    remin))
  end;
function yprim(y:Real): Integer;
  begin
    yprim:=round((getmaxy-10)*(y-immax)/(immin-
    immax))
  end;
function citeste_bit(k:Longint;poz:Byte): Byte;
  const 1: longint=1;
  begin
    if k and(1 Shl (poz-1))=0 then citeste_bit:=0
                              else citeste_bit:=1
```

```
    end;
function pow(k:longint): Real;
  begin
    if odd(k) then pow:=-exp(k*ln(0.25))
              else pow:=exp(k*ln(0.25))
  end;
procedure putere(k:Integer; var rez:complex);
  var  cit, rest :Integer;
       p: Real;
  begin
     cit:=-k div 4; rest:=-k mod 4; p:=pow(cit);
  case rest of
  0: begin rez.re:= p; rez.im:=0 end;
  1: begin rez.re:=p*(-0.5); rez.im:=p*(-0.5) end;
  2: begin rez.re:=0; rez.im:=p*0.5 end;
  3: begin rez.re:=p*0.25; rez.im:=p*-0.25 end
  end
end;

begin {main}
 Write('nr. cifre zecimale(maxim ',NrCfMax,'): ');
 Readln(n);
 if n>NrCfMax then begin Writeln('prea multe
 cifre'); Readln; Exit end;
 Write('nr. de figuri congruente(putere a lui 2):
 '); Readln(nfig);
 gd:=0; Initgraph(gd, gm, 'c:\bp\bgi');
 if Graphresult<>0 then begin Write('eroare
 initializare'); Readln; Exit end;
 remin:=-1; remax:=1; immin:=-1; immax:=1;
     {fereastra afisare}
 Setcolor(8);
 Moveto(xprim(remin), yprim(0));
     {linii, axe}
 Lineto(xprim(remax), yprim(0));
 Moveto(xprim(0), yprim(immin));
Lineto(xprim(0), yprim(immax));
 Moveto(xprim(remin), yprim(immin));
Lineto(xprim(remax), yprim(immin));
 Moveto(xprim(remax), yprim(immax));
Lineto(xprim(remin), yprim(immax));
 Moveto(xprim(remin), yprim(immin));
Lineto(xprim(remin), yprim(immax));
 Moveto(xprim(remax), yprim(immax));
Lineto(xprim(remin), yprim(immax));
 i:=0;
 while (i<exp(n*ln(2))) and not Keypressed do
  begin
    sum.re:=0; sum.im:=0;
    for j:=1 to n do
       if citeste_bit(i,j)=1 then begin
                        putere(-j, z);
                        sum.re:=sum.re+z.re;
                        sum.im:=sum.im+z.im;
                      end;
    if nfig<=16 then cul:=i mod nfig
      {calcul pt. culoare pixel}
       else cul:=(i mod nfig) div (nfig div 16);
    if cul=0 then cul:=8;
    Putpixel(xprim(sum.re), yprim(sum.im), cul);
     {desenarea unui punct al fig.}
   Inc(i);
  end;
Outtextxy(0, 0, 'gata');
Readln
end.
```



"Dragonul cu două capete" (n=20, nfig=32)

Desenele cu care este ilustrat articolul au fost realizate cu acest program, rulat cu următoarele date de intrare:

fig.1: n=18, nfig=1;

fig.2: n=20, nfig=32.

Indicații bibliografice:

În încheiere, recomand celor interesați următoarele lucrări:

(1) D. E. Knuth-"*Tratat de programare a calculatoarelor*"-volumul "Algoritmi seminumerici";

(2) M. Vlada, A. Posea ș.a.-"*Grafică pe calculator în limbajele Pascal și C*"

(3) Dick Oliver - "*Fractali*"

Recomand, de asemenea, celor care au acces la Internet, parcurgerea *fractal-faq* (fișierul cu întrebări și răspunsuri frecvente în domeniul fractalilor), mai ales pentru faptul că prezintă numeroase adrese de servere cu informații din acest domeniu. Fractal-faq poate fi luat printr-un browser www, prin ftp sau lynx de la numeroase adrese, de exemplu:

ftp://rtfm.mit.edu/pub/usenet/news.answers/fractal-faq [18.181.0.24]

byrd.mu.wvnet.edu/pub/estepp/fracha/fracta1.faq [129.71.32.152]

Domnul Stelian Ciurea este profesor la Sibiu și poate fi contactat prin e-mail la adresa ciurea@vectra.sibiu.ro.

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie*

... un MANUAL.

Această carte este concepută în conformitate cu Programa analitică a clasei a XI-a, profilul Informatică. Ea poate fi considerată o introducere elementară în limbajul C. Fiecare noțiune este explicată în mod clar și utilizată în programe simple. De asemenea, capitolele sunt însoțite de exerciții propuse care contribuie la fixarea și înțelegerea noțiunilor întâlnite în capitolele respective.

Cartea se adresează atât elevilor cât și celor interesați să se inițieze în limbajul C care în prezent este unul dintre cele mai utilizate limbaje de programare.

Liviu Negrescu este cercetător principal gr. I la societatea SOFTWARE ITC S.A., filiala Cluj, unde se ocupă cu dezvoltări de programe de sistem și aplicative.
În același timp, desfășoară o activitate didactică prestigioasă de peste 15 ani, la Facultatea de Calculatoare de la Universitatea Tehnică din Cluj.
În decursul anilor a fost titular a mai multe cursuri, ca de exemplu:
- Limbaje de programare și compilatoare;
- Proiectarea translatoarelor etc.

Liviu Negrescu are o serie de lucrări științifice publicate în diferite reviste de specialitate. În colaborare cu alți cercetători de la SOFTWARE ITC S.A. a implementat sisteme cu aplicații în diferite domenii:
- Construcția de compilatoare;
- Interfețe utilizator;
- Prelucrări de date din domenii economice și medicale.

Liviu Negrescu este autorul mai multor cărți apărute în decursul anilor, cu privire la diferite limbaje de programare (CPL1-subset al limbajului PL/1, CDL-limbaj de descriere al compilatoarelor precum și referitoare la limbajele C și C++. Lucrarea beneficiază de experiența vastă a autorului câștigată atât în activitatea de cercetare cât și în cea didactică.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Numere FIBONACCI

de Emanuela Mateescu

Cum arăta oare Europa secolului XII ? Dacă vă imaginați doar bezna unui sălbatic Ev Mediu timpuriu, vă înșelați. În acea epocă se scriau tratate de știință și, se pare, se organizau concursuri de matematică. Numele împăraților epocii s-au pierdut, dar numele matematicianului Fibonacci a rămas...

Cine a fost Fibonacci?

Leonardo din Pisa, cunoscut sub numele Fibonacci[1], a fost unul dintre cei mai mari matematicieni europeni ai Evului Mediu.

Născut în Italia, în 1175, a fost educat în Nordul Africii, unde tatăl său deținea un post diplomatic. În 1202 revine în Italia și publică un tratat de aritmetică și algebră intitulat "*Liber abaci*". În acest tratat introduce pentru prima dată în Europa sistemul de numerație pozițional arab. De asemenea, în 1220 publică "*Practica geometriae*", un compendiu de rezultate din geometrie și trigonometrie, iar în 1225 "*Liber quadratorum*", în care studia calculul radicalilor cubici.

Totuși, *Fibonacci* a rămas în memoria noastră prin binecunoscutul șir Fibonacci:

0, 1, 1, 2, 3, 5, 8, 13, ...

Problema iepurilor

Se pare că și pe vremea lui *Fibonacci* se organizau concursuri de matematică. În *Pisa*, a participat și *Fibonacci* la un astfel de concurs care a fost condus de însuși împăratul *Frederik II*. Problema propusă concurenților suna astfel:

Plecând de la o singură pereche de iepuri și știind că fiecare pereche de iepuri produce în fiecare lună o nouă pereche de iepuri, care devine "productivă" la vârsta de o lună, calculați câte perechi de iepuri vor fi după n luni.

Soluție:

Să notăm f_n numărul de perechi de iepuri după n luni. Numărul de perechi de iepuri după n+1 luni, notat f_{n+1}, va fi f_n (iepurii nu mor niciodată!) la care se adaugă iepurii nou-născuți. Dar iepurașii se nasc doar din perechi de iepuri care au cel puțin o lună, deci vor fi f_{n-1} perechi de iepuri nou-născuți.

Obținem astfel o relație de recurență:

$$\begin{aligned} f_{n+1} &= f_n + f_{n-1}, \\ f_1 &= 1; \\ f_0 &= 0. \end{aligned} \qquad (1)$$

Această relație de recurență reprezintă regula care generează termenii șirului *Fibonacci*[2].

Exerciții:

1. Scrieți o funcție recursivă și o funcție nerecursivă care să calculeze f_n. Comparați cele două funcții din punctul de vedere al complexității spațiu și al complexității timp.

2. Scrieți un program care să afișeze numărul de posibilități în care un om poate urca o scara cu n ($n \leq 256$) trepte, știind că la un pas, el poate urca o singură treaptă sau cel mult două.

3. Gazda unei reuniuni la care sunt prezenți și profesori știe că atunci când doi profesori se întâlnesc inevitabil discută (ce plicticos!!!) despre școală. Scrieți un program care să afișeze numărul de posibilități în care putem așeza n dintre persoanele de la reuniune pe un rând de n scaune, astfel încât să nu existe doi profesori așezați pe scaune alăturate.

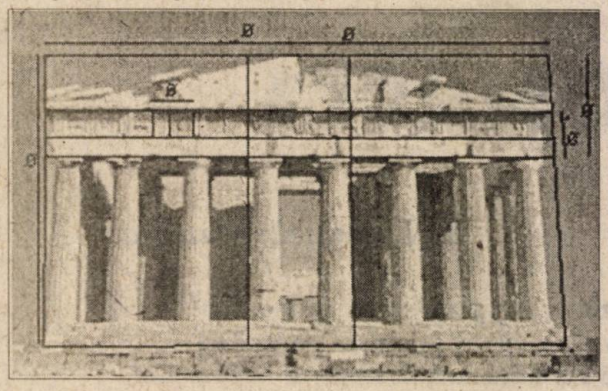
4. Scrieți un program care să afișeze numărul de posibilități de a achita o sumă dată S, folosind numai monede de *1*, respectiv *2 lei*.

Secțiunea de aur

Problema înmulțirii iepurilor nu este prea realistă. Dar șirul generat are numeroase aplicații, deosebit de interesante. Unul dintre cele mai interesante aspecte este legătura dintre numerele *Fibonacci* și *secțiunea de aur*.

Secțiunea de aur este probabil unul dintre cele mai misterioase numere, constituind de secole o fascinație pentru matematicieni și artiști. Ca și numerele iraționale π sau e, pare a face parte din "constituția" Universului, secțiunea de aur regăsindu-se sistematic în lumea vie. De

exemplu, o regăsim în modul de dispunere a frunzelor, petalelor sau a semințelor la plante, în raportul dintre diferite părți ale corpului omenesc.



Acest număr a fost cunoscut și studiat încă din antichitate, sculptura și arhitectura Greciei antice din secolul lui *Pericle* respectând cu rigurozitate secțiunea de aur, aceasta fiind considerată o măsură a armoniei și echilibrului.

O modalitate de determinare a secțiunii de aur

Să considerăm un segment AB și $C \in AB$, un punct care împarte segmentul AB în două părți inegale (AC > CB). Segmentele AC și CB sunt în *secțiunea* (raportul) *de aur* (notat în mod uzual cu φ) dacă:

$$\frac{AB}{AC} = \frac{AC}{CB}$$

Notând AC cu x și CB cu y, obținem:

$$\frac{x+y}{x} = \frac{x}{y} = \varphi \Leftrightarrow 1 + \frac{y}{x} = \frac{x}{y} = \varphi \Leftrightarrow$$

$$\Leftrightarrow 1 + \frac{1}{\varphi} = \varphi \Leftrightarrow \varphi^2 - \varphi - 1 = 0$$

Rezolvând ecuația de gradul al II-lea obținută și luând în considerare numai rădăcina pozitivă, obținem:

$$\varphi = \frac{1+\sqrt{5}}{2} = 1{,}61803398875\ldots$$

Plecând de la relația $1+\varphi=\varphi^2$,
prin înmulțiri succesive cu φ obținem:
$\varphi+\varphi^2=\varphi^3$
$\varphi^2+\varphi^3=\varphi^4$
...
$\varphi^{n-1}+\varphi^n=\varphi^{n+1}$.

Prin urmare, șirul $1, \varphi, \varphi^2, \varphi^3, \ldots, \varphi^n, \ldots$ formează o progresie geometrică cu rația φ, dar în plus respectă și o lege aditivă (fiecare termen este egal cu suma celor doi termeni precedenți).

Să revenim acum la relația de recurență *(1)*, care definește șirul *Fibonacci*. Pentru a determina o altă formulă pentru termenii șirului *Fibonacci*, vom determina α, β, u, v astfel încât $f_n = \alpha \cdot u^n + \beta \cdot v^n$, $\forall n \geq 0$.

Deci dorim să scriem f_n ca suma termenilor generali a două progresii geometrice. Folosind relația *(1)*, obținem:

$$\alpha \cdot u^n + \beta \cdot v^n = \alpha \cdot u^{n-1} + \beta \cdot v^{n-1} + \alpha \cdot u^{n-2} + \beta \cdot v^{n-2} =$$
$$= \alpha \cdot u^{n-2} \cdot (u^2 - u - 1) + \beta \cdot v^{n-2} \cdot (v^2 - v - 1) \ \forall n > 1.$$

Deci u și v sunt rădăcinile ecuației $x^2-x-1=0$. Impunând condițiile ca $x_0 = 0$ și $x_1 = 1$, obținem:

$$f_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right], \forall n \geq 0.$$

Deci:

$$f_n = \frac{\varphi^n - \left(-\frac{1}{\varphi}\right)^n}{\sqrt{5}} = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}} \qquad (2)$$

Interesant este că, deși în formula (2) intervin numere iraționale, rezultatul este întotdeauna un număr natural. Din relația (2) putem deduce o formulă mai simplă pentru f_n, observând că $1/\varphi$ este un număr subunitar, puterile sale tinzând rapid către 0.

Obținem o formulă mult mai simplă pentru f_n:

$$f_n = \left[\frac{\varphi^n}{\sqrt{5}} \right] \qquad (3)$$

unde $[x]$ reprezintă numărul întreg cel mai apropiat de x.

Observație:

Plecând de la formula $\varphi = 1 + 1/\varphi$, deducem o altă formulă de definire a numărului irațional φ:

$$\varphi = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{\ddots}}}}$$

Sume de numere Fibonacci

$$f_0 + f_2 + f_4 + \ldots + f_{2k} = f_{2k+1} - 1 \qquad (4)$$

Demonstrație:

$$f_0 + f_2 + f_4 + \ldots + f_{2k} = -1 + (f_1 + f_2) + f_4 + \ldots + f_{2k} =$$
$$= -1 + (f_3 + f_4) + f_6 + \ldots + f_{2k} = \ldots =$$
$$= -1 + f_{2k-1} + f_{2k} = f_{2k+1} - 1.$$

Exerciții:

- Demonstrați că:

$$f_1 + f_3 + f_5 + \ldots + f_{2k-1} = f_{2k} \qquad (5)$$

- Demonstrați prin inducție că:

$$f_1^2 + f_2^2 + f_3^2 + \ldots + f_k^2 = f_k \cdot f_{k+1} \qquad (6)$$

Sistemul de numerație Fibonacci

Cu ajutorul termenilor șirului *Fibonacci* putem defini un sistem de numerație pozițional foarte interesant. Mulțimea simbolurilor care reprezintă cifrele acestui sistem de numerație este {0, 1}, deci baza sistemului de numerație este 2. În sistemele de numerație poziționale uzuale fiecare cifră dintr-un număr valorează de b ori mai mult decât cifra din dreapta sa (unde b reprezintă baza sistemului de numerație), deci cifrele numărului, de la dreapta spre stânga, corespund pozițional puterilor bazei sistemului de numerație.

$$\overline{a_n a_{n-1} \ldots a_1 a_0}(b) = a_n b^n + a_{n-1} b^{n-1} + \ldots + a_1 b + a_0$$

Prima regulă de reprezentare a numerelor naturale în sistemul de numerație *Fibonacci* este de a asocia pozițional cifrelor numărului termeni ai șirului Fibonacci, începând cu f_1, și nu puteri ale bazei.

$$\overline{a_n a_{n-1} \ldots a_1} = a_n f_n + a_{n-1} f_{n-1} + \ldots + a_1 f_1$$

Pentru ca reprezentarea numerelor naturale în sistemul *Fibonacci* să fie unică impunem condiția ca oricare două cifre consecutive să nu fie egale cu 1. Condiția este naturală deoarece o succesiune de forma ...011... poate fi înlocuită cu ...100... .

Exemple:

Sistemul zecimal	Sistemul *Fibonacci*
1	1
2	10
3	100
4	101
5	1000
6	1001
7	1010
8	10000

Aplicație:

Conversia numerelor naturale din sistemul de numerație zecimal în sistemul de numerație *Fibonacci* .

Listing CONVERS.PAS

```
{$N+}
Program Conversie_Zecimal_Fibonacci;
var n,fn,fn_1,f:comp;
Begin
  Write('Introduceti un numar natural in baza 10');
  Readln(n);
  { determina doi termeni consecutivi din sirul  }
  { Fibonacci, mai mari decat n                  }
  fn:=1; fn_1:=1;
  repeat
    f:=fn_1+fn;
    fn_1:=fn;
    fn:=f;
  until fn_1>n;
  { afiseaza pe ecran reprezentarea numarului x  }
  { in sistemul de numeratie Fibonacci           }
  Write('Numarul ',n:10:0,' in reprezentarea
  Fibonacci este ');
  if n=0
  then Write(0)
  else
    while fn_1>1 do
    begin
      f:=fn-fn_1;
      fn:=fn_1;
      fn_1:=f;
      if f>n
      then Write(0)
      else
        begin
          Write(1);
          n:=n-f
        end
    end;
    Writeln;
    Readln
End.
```

Observație:

Condiția ca reprezentarea *Fibonacci* să nu conțină doi de 1 consecutivi nu este verificată în mod explicit. Se poate demonstra prin inducție după p, numărul de cifre din reprezentarea *Fibonacci*, că, dacă cifra x_k este egală cu 1, atunci x_{k-1} este 0, $\forall\, k \in \{p, p\text{-}1, \ldots, 1\}$.

Exercițiu:

Scrieți un program de conversie a numerelor naturale din sistemul de numerație *Fibonacci* în sistemul de numerație zecimal.

Aplicație:

Jocul Tziansitzi (*Concursul de Informatică prin E-mail*, Oradea, 1997)

Jocul *Tziansitzi*, ceea ce în traducere înseamnă "*alegerea pietrelor*", este un joc popular din China. Doi jucători extrag alternativ pietre din două grămezi. La fiecare mișcare un jucător poate extrage același număr de pietre din ambele grămezi sau un număr oarecare de pietre din una din grămezi. Câștigă jucătorul care ia și ultima piatră. Se cere să testați dacă, pentru o configurație dată, primul jucător are sau nu strategie de câștig și dacă da să afișați o mutare a primului jucător astfel încât, după această mutare, al doilea jucător să fie în situație clară de pierdere (orice strategie ar adopta, al doilea jucător nu poate câștiga, în ipoteza că primul jucător nu face greșeli).

Restricții de intrare:

Numele fișierului de intrare se citește de la tastatură.

Fișierul de intrare conține o singură linie de forma:

n m

unde n = numărul de pietre din prima grămadă;

m = numărul de pietre din a doua grămadă.

(n, m sunt numere naturale mai mici decât 1.0E18; pentru simplitate, presupunem $n \le m$).

Restricții de ieșire:

Numele fișierului de ieșire se citește de la tastatură.

Fișierul de ieșire poate conține mesajul 'Jucatorul 1 este in situatie de pierdere' sau

— pe prima linie mesajul 'Jucatorul 1 are strategie de castig';

— pe a doua linie x y, unde x reprezintă numărul de pietre extrase din prima grămadă, iar y numărul de pietre extrase din a doua grămadă de primul jucător la o mutare prin care al doilea jucător va fi plasat într-o situație de pierdere;

— pe a treia linie n1 m1, numărul de pietre rămase în prima grămadă, respectiv numărul de pietre rămase în cea de a doua grămadă după mutarea primului jucător.

Timp maxim de executare/test:

0.5 s (indiferent de calculator).

Exemplul 1:

Dacă fișierul de intrare conține:

```
1 2
```

În fișierul de ieșire vom obține:

```
Jucatorul 1 este in situatie de pierdere
```

Exemplul 2:

Dacă fișierul de intrare conține:

```
5 6
```

În fișierul de ieșire vom obține:

```
Jucatorul 1 are strategie de castig
0 3
5 3
```

Soluție

Definiție:

Perechea (n, m), cu $n<m$ se numește distinsă dacă:

1. reprezentarea *Fibonacci* a lui n se termină cu un număr par de zerouri;

2. reprezentarea *Fibonacci* a lui m se obține din reprezentarea *Fibonacci* a lui n prin adăugarea la sfârșit a unui 0.

Exemple:

1. $n = 1$ (1, în sistemul *Fibonacci*), $m = 2$ (10, în sistemul *Fibonacci*)
2. $n = 3$ (100, în sistemul *Fibonacci*), $m = 5$ (1000, în sistemul *Fibonacci*)
3. $n = 4$ (101, în sistemul *Fibonacci*), $m = 7$ (1010, în sistemul *Fibonacci*)

Propoziție:

Orice număr natural aparține unei singure perechi distinse.

Demonstrație:

Dacă reprezentarea numărului în sistemul de numerație Fibonacci se termină cu un număr par de zerouri, al doilea număr din pereche se obține adăugând un 0 final, altfel al doilea număr din pereche se obține prin suprimarea unui 0 final.

Propoziție:

În șirul $\{m - n \mid (n, m)$ pereche distinsă$\}$ fiecare număr natural apare o singură dată.

Demonstrație:

Fie $d \in \mathbb{N}$, iar $x_p x_{p-1} \ldots x_1$, reprezentarea *Fibonacci* a lui d.

$$d = x_p x_{p-1} \ldots x_1 = x_p \cdot f_p + x_{p-1} \cdot f_{p-1} + \ldots + x_1 f_1$$

Dacă reprezentarea Fibonacci a lui d se termină cu un număr impar de zerouri, atunci

$$n = x_p x_{p-1} \ldots x_1 0 = x_p \cdot f_{p+1} + x_{p-1} \cdot f_p + \ldots + x_1 \cdot f_2 \qquad (7)$$

$$m = x_p x_{p-1} \ldots x_1 00 = x_p \cdot f_{p+2} + x_{p-1} \cdot f_{p+1} + \ldots + x_1 \cdot f_3 \qquad (8)$$

$$m - n = x_p \cdot (f_{p+2} - f_{p+1}) + x_{p-1} \cdot (f_{p+1} - f_p) + \ldots + x_1 \cdot (f_3 - f_2) =$$

$$= x_p \cdot f_p + x_{p-1} \cdot f_{p-1} + \ldots + x_1 f_1 = d$$

Dacă reprezentarea *Fibonacci* a lui d se termină cu un număr par de zerouri, deci $x_1 = x_2 = x_{2k} = 0$, $x_{2k+1} \neq 0$, atunci:

$$n = x_p x_{p-1} \ldots x_{2k+2} \underbrace{0101 \ldots 01}_{01 \text{ de } k+1 \text{ ori}} = x_p \cdot f_{p+1} + x_{p-1} \cdot f_p + \ldots + x_{2k+2} \cdot f_{2k+3} + (f_{2k+1} + f_{2k-1} + \ldots + f_3 + f_1)$$

$$m = x_p x_{p-1} \ldots x_{2k+2} \underbrace{0101 \ldots 01}_{01 \text{ de } k+1 \text{ ori}} 0 = x_p \cdot f_{p+2} + x_{p-1} \cdot f_{p+1} + \ldots + x_{2k+2} \cdot f_{2k+4} + (f_{2k+2} + f_{2k} + \ldots + f_4 + f_2)$$

$$m - n = x_p \cdot (f_{p+2} - f_p + 1) + x_{p-1} \cdot (f_{p+1} - f_p) + \ldots + x_{2k+2} \cdot (f_{2k+4} - f_{2k+3}) + (f_{2k+2} - f_{2k+1}) + \ldots + (f_2 - f_1) = x_p \cdot f_p + x_{p-1} \cdot f_{p-1} + \ldots + x_{2k+2} f_{2k+2} + f_{2k+1} = d,$$ conform proprietății (3).

Unicitatea se poate demonstra, aplicând formulele *(7)*, *(8)* în sens invers (formula *(7)* corespunde cazului în care n se termină cu 0, formula *(8)*, pentru cazul în care n nu se termină cu 0).

Teoremă:

Dacă n și m, numărul de pietre din prima, respectiv cea de a doua grămadă nu formează o pereche distinsă, atunci primul

jucător are strategie sigură de câștig (câștigă dintr-o singură mutare, sau poate aduce jocul printr-o singură mutare într-o situație în care (n, m) formează o pereche distinsă.

Schița de demonstrație [3]

1. Dacă $n = m$ sau $n = 0$ primul jucător câștigă dintr-o singură mutare.

2. Dacă $0 < n < m$, determinăm perechea distinsă (n, p) și perechea distinsă (a, b) astfel încât $b-a=m-n$. Dacă $p<m$, atunci se poate trece cu o singură mutare de la perechea (n, m) la perechea distinsă (n, p). Dacă $p > m$, deducem $p-n> b-a=m-n$, deci perechea distinsă (n, p) este "mai mare" decât perechea distinsă (a, b), deci $p>m$, $a>n$. Deducem $n-a=m-b$, deci dintr-o singură mișcare putem trece la perechea distinsă (a, b).

3. Dacă perechea (n, m) este distinsă, după orice mișcare ea va înceta să mai fie distinsă.

Listing TZ.PAS

```
{$N+}
Program TZ;
const NrMaxPietre=1E18;
      DimMax=87;
type Indice=0..DimMax+2;
     ReprezentareF=array[Indice] of 0..1;
     SirFibonacci=array[Indice] of comp;
var n,m,p,a,b:comp;
    s:SirFibonacci;
    rn,rm:ReprezentareF;
    fout:Text; nume:string;

  procedure GenerareSirFibonacci;
  { genereaza in vectorul s termenii sirului      }
  { Fibonacci necesari pentru a reprezenta in     }
  { sistemul de numeratie Fibonacci numerele pana }
  { la 1E18                                       }
  var i:Indice;
  begin
    s[1]:=1; s[2]:=2;
    for i:=3 to DimMax+2 do
      s[i]:=s[i-1]+s[i-2]
  end;

  procedure Reprezinta(x:comp;
  var rx:ReprezentareF);
  { determina in rx reprezentarea numarului x in  }
  { sistemul de numeratie Fibonacci               }
  var i:Indice;
  begin
    i:=DimMax;
    while i>0 do
    begin
      if s[i]>x
        then rx[i]:=0
        else begin rx[i]:=1; x:=x-s[i] end;
      Dec(i)
    end
  end;

  procedure Citire;
  var f:Text; nume:string;
  begin
    Write('fisier intrare='); Readln(nume);
    Assign(f,nume); Reset(f);
    Readln(f,n,m);
    Close(f)
  end;

  function NrZero(var rx:ReprezentareF):Indice;
  { returneaza numarul de zerouri "finale"        }
  { ale reprezentarii rx                          }
  var i:Indice;
  begin
    i:=1;
    while (i<=DimMax) and (rx[i]=0) do Inc(i);
    NrZero:=i-1
  end;

  function Test:Boolean;
  { returneaza true daca numerele n, m formeaza o }
  { pereche distinsa (reprezentarea Fibonacci a   }
  { lui n are un numar par de zerouri, iar        }
  { reprezentarea Fibonacci a lui m se obtine     }
  { adaugand un zero la reprezentarea lui n)      }
  var n0,m0,i:Indice;
      egal:Boolean;
  begin
    Test:=false;
    n0:=NrZero(rn); m0:=NrZero(rm);
    if (n0 mod 2=0) and (m0=n0+1)
    then
      begin
        i:=n0+1; egal:=true;
        while (i<=DimMax) and egal do
        begin
          if rn[i]<>rm[i+1] then egal:=false;
          Inc(i)
        end;
        Test:=egal
      end
  end;

  function DeterminaTermen(var rx: ReprezentareF):
                                  comp;
  { returneaza numarul natural care formeaza o    }
  { pereche distinsa cu numarul cu                }
  { reprezentarea rx                              }
  var x0,i:Indice; v:comp;
  begin
    v:=0;
    x0:=NrZero(rx);
```

```
    d0 mod 2=0
  then for i:=1 to DimMax do v:=v+s[i+1]*rx[i]
  else for i:=2 to DimMax do v:=v+s[i-1]*rx[i];
  DeterminaTermen:=v
end;

procedure DeterminaPereche(d:comp; var a,b:comp);
{ determina unica pereche distinsa (a,b),       }
{ astfel incat b-a=d                            }
var rd:ReprezentareF;
    d0,i,m:Indice;
begin
  Reprezinta(d,rd);
  rd[DimMax+1]:=0; rd[DimMax+2]:=0;
  d0:=NrZero(rd);
  a:=0; b:=0;
  if d0 mod 2=1
  then
    for i:=1 to DimMax do
    begin
      a:=a+rd[i]*s[i+1];
      b:=b+rd[i]*s[i+2]
    end
  else
    begin
      m:=d0 div 2;
      for i:=0 to m do
      begin
        a:=a+s[2*i+1];
        b:=b+s[2*i+2]
      end;
      for i:=2*m+2 to DimMax do
      begin
        a:=a+rd[i]*s[i+1];
        b:=b+rd[i]*s[i+2]
      end
    end
end;

Begin
  Citire;
  GenerareSirFibonacci;
  Reprezinta(n,rn);
  Reprezinta(m,rm);
  Write('fisier iesire='); Readln(nume);
  Assign(fout,nume); Rewrite(fout);
  if Test
  then
    Writeln(fout,
     'Jucatorul 1 este in situatie de pierdere')
  else
    begin
      Writeln(fout,'Jucatorul 1 are strategie de
              castig');
      if (n=m) or (n=0)
      then
        begin
          Writeln(fout,n:20:0,m:20:0);
          Writeln(fout,n-n:20:0,' ',m-m:20:0)
        end
      else
        begin    { 0<n<m }
          p:=DeterminaTermen(rn);
          if m>p
          then
            begin
              Writeln(fout,0,' ',m-p:20:0);
              Writeln(fout,n:20:0,p:20:0)
            end
          else
            begin
              DeterminaPereche(m-n,a,b);
              Writeln(fout,n-a:20:0,m-b:20:0);
              Writeln(fout,a:20:0,b:20:0)
            end
        end
    end;
  Close(fout)
End.
```

Observație:

O implementare mai simplă se poate obține luând în considerare relația dintre numerele *Fibonacci* și φ, secțiunea de aur.

Ideea este de a genera șirul perechilor distinse (n_k, m_k), folosind relațiile:

$$m_k = \left[\frac{3+\sqrt{5}}{2}k\right] \qquad n_k = \left[\frac{1+\sqrt{5}}{2}k\right]$$

Propunem ca exercițiu demonstrația acestor formule.

Bibliografie:

Câmpan, Florica- *Probleme celebre*. Editura Albatros, 1972.

Iaglom, A.M.; Iaglom, I.M.- *Probleme neelementare tratate elementar*. Editura Tehnică, 1983.

Pagina Internet a Universității Surrey, Guildford, Surrey, Marea Britanie.

Note:

1) O prescurtare de la *filius Bonacci* (fiul lui *Bonacci*).

2) Acest șir a fost denumit **șirul Fibonacci** de către matematicianul francez *Edouard Lucas* (1842-1891).

3) O demonstrație riguroasă se găsește în Iaglom, A.M.; Iaglom, I.M.- *Probleme neelementare tratate elementar*. Editura Tehnică, 1983.

D-na Emanuela Mateescu este profesoară la Liceul de Informatică "Grigore Moisil" din Iași

PRINCIPIUL inducției în informatică

Doru Popescu Anastasiu

Există multe probleme care pot fi rezolvate prin inducție matematică. În general unei astfel de probleme putem să-i asociem un algoritm. Demonstrarea prin inducție a corectitudinii anumitor rezultate permit obținerea sigură a unei soluții cu ajutorul unui algoritm, pentru date concrete.

Pentru a ilustra cele afirmate, vom prezenta în continuare două probleme care folosesc această idee. Prima se poate demonstra prin inducție matematică cu pasul 3, iar a doua cu pasul 4.

Problema 1: Partiționarea unui pătrat

Fiind cunoscut numărul natural, $n \geq 6$ *să se împartă un pătrat în* n *pătrățele.*

Exemplu:

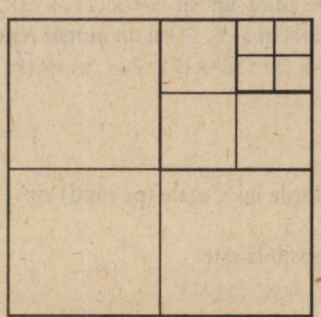
Pentru n=10, o soluție posibilă este:

```
1 1 1 1 4 4 7 8
1 1 1 1 4 4 10 9
1 1 1 1 5 5 6 6
1 1 1 1 5 5 6 6
2 2 2 2 3 3 3 3
2 2 2 2 3 3 3 3
2 2 2 2 3 3 3 3
2 2 2 2 3 3 3 3
```

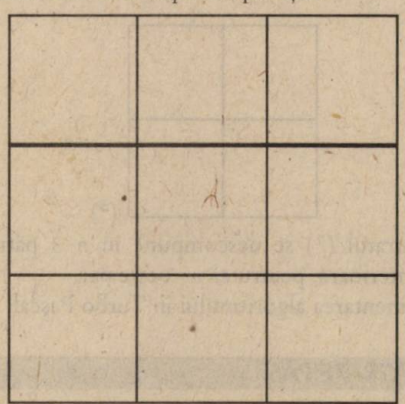
iar pentru n=9, avem:

```
1 2 3
4 5 6
7 8 9
```

Primul tablou corespunde partiției:



Al doilea tablou corespunde partiției:



Observăm mai întâi că datorită principiului inducției matematice problema are soluție pentru orice $n \geq 6$. Pasul inducției este 3.

Astfel problema poate fi rezolvată construind succesiv un șir de tablouri.

Primele 3 descompuneri sunt:

Pentru n=6

```
1 2 3
6 6 4
6 6 5
```

Pentru n=7

```
1 1 4 5
1 1 7 6
2 2 3 3
2 2 3 3
```

Pentru n=8

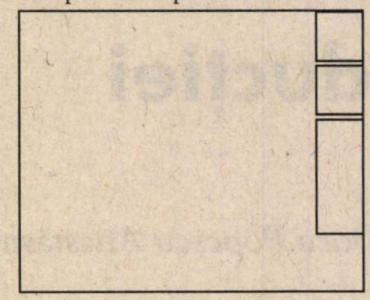
```
1 2 3 4
8 8 8 5
8 8 8 6
8 8 8 7
```

Observație:

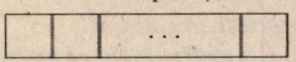
Soluția nu este unică!

În algoritmul pe care îl prezentăm se iau în considerare cazurile:

- dacă n este par atunci pătratul va fi descompus astfel:

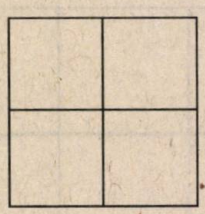


(n/2)-1 pătrățele



(n/2)-1 pătrățele

- dacă n este impar atunci pătratul va fi descompus astfel:



(*)

iar pătratul (*) se descompune în n-3 pătrățele cu metoda anterioară, pentru că n-3 este par.

Implementarea algoritmului în Turbo Pascal:

Listing PATRATE.PAS

```
Program patrate;
uses Crt;
type tabl=array[1..50,1..50] of Word;
var jum,dim,n,i,j,k:Integer;
    a:tabl;

procedure afisare(a:tabl; dim:Integer);
var i,j:Integer;
begin
  for i:=1 to dim do
  begin
    for j:=1 to dim do Write(a[i,j]:5);
    Writeln
  end
end;

Begin
  Clrscr;
  Writeln('Introduceti numarul de patratele ');
  Readln(n);
  for i:=1 to 50 do
    for j:=1 to 50 do a[i,j]:=1;
  if n mod 2=0
  then
    begin
      dim:=n div 2;
      k:=1;
      for i:=1 to dim do
      begin
        Inc(k);
        a[i,dim]:=k
      end;
      k:=k-1;
      for i:=1 to dim do
      begin Inc(k); a[dim,i]:=k
      end;
      afisare(a,dim)
    end
  else
     begin
       dim:=n-3;
       jum:=dim div 2;
       for i:=1 to jum do
         for j:=jum+1 to dim do a[j,i]:=2;
       for i:=jum+1 to dim do
         for j:=1 to jum do a[j,i]:=3;
       for i:=jum+1 to dim-1 do
         for j:=jum+1 to dim-1 do a[i,j]:=4;
       k:=4;
       for i:=jum+1 to dim do
       begin
         Inc(k);
         a[i,dim]:=k
       end;
       for i:=jum+1 to dim-1 do
       begin
         Inc(k);
         a[dim,i]:=k
       end;
       afisare(a,dim)
  end;
 Readln
End.
```

Problema 2: Combinații de + și -

Pentru un număr natural n, să se găsească o combinație de semne + și - (adică un șir x=(x(1),x(2),...,x(k)), unde x(i) se află în {-1,1}) și un număr natural k nenul, astfel încât: n=x(1)*12+x(2)*22+...+x(k)*k2.

Exemplu:

Pentru valorile lui n egale (pe rând) cu:

2 4 8 5

o soluție posibilă este:

- - - +

- - +

```
- - + + - - +
+ + - - +
```

Problema sub formă algebrică (ce aparține lui P. Erdos și M. Suranyi) se rezolvă construind iterativ combinații de semne pentru numerele consecutive de la 0 la numărul n. Astfel avem o inducție matematică cu pasul 4, primii 4 termeni fiind:

$0=1^2+2^2-3^2+4^2-5^2-6^2+7^2$

$1=1^2$

$2=-1^2-2^2-3^2+4^2$

$3=-1^2+2^2$

$4=-1^2-2^2+3^2$,

celelalte combinații se construiesc inductiv cu pasul 4, folosind relația:

$4=(m+1)^2-(m+2)^2-(m+3)^2+(m+4)^2$.

Programul care rezolvă această problemă pentru n citit de la tastatură:

Listing PLUS_MIN.PAS

```
Program plus_si_minus;
uses Crt;
type vector=array[1..10000] of -1..1;
var x,y,z,u:vector;
    rest:Byte; i,n,dimx,dimy,dimz,dimu,s:Longint;

procedure afisare_verificare(v:vector;
dim:Integer);
var i,s:Longint;
begin
  Writeln('Sirul de semne este ');
  for i:=1 to dim do
    if v[i]=-1
    then Write('- ')
    else Write('+ ');
  Writeln;
  s:=0;
  for i:=1 to dim do s:=s+v[i]*i*i;
  Writeln(n,' = ',s)
end;

Begin
  Clrscr;
  Writeln('Introduceti un nr. natural <10000');
  Write('n='); Readln(n);
  dimx:=7;
  x[1]:=1; x[2]:=1; x[3]:=-1; x[4]:=1;
  x[5]:=-1; x[6]:=-1; x[7]:=1;
  dimy:=1;
  y[1]:=1;
  dimz:=4;
  z[1]:=-1; z[2]:=-1; z[3]:=-1; z[4]:=1;
  dimu:=2;
  u[1]:=-1; u[2]:=1;
  for i:=4 to n do
  begin
    rest:=i mod 4;
    case rest of
    0:begin
       dimx:=dimx+1; x[dimx]:=1;
       dimx:=dimx+1; x[dimx]:=-1;
       dimx:=dimx+1; x[dimx]:=-1;
       dimx:=dimx+1; x[dimx]:=1
      end;
    1:begin
       dimy:=dimy+1; y[dimy]:=1;
       dimy:=dimy+1; y[dimy]:=-1;
       dimy:=dimy+1; y[dimy]:=-1;
       dimy:=dimy+1; y[dimy]:=1
      end;
    2:begin
       dimz:=dimz+1; z[dimz]:=1;
       dimz:=dimz+1; z[dimz]:=-1;
       dimz:=dimz+1; z[dimz]:=-1;
       dimz:=dimz+1; z[dimz]:=1
      end;
    3:begin
       dimu:=dimu+1; u[dimu]:=1;
       dimu:=dimu+1; u[dimu]:=-1;
       dimu:=dimu+1; u[dimu]:=-1;
       dimu:=dimu+1; u[dimu]:=1
      end
    end
  end;
  rest:=n mod 4;
  case rest of
  0: afisare_verificare(x,dimx);
  1: afisare_verificare(y,dimy);
  2: afisare_verificare(z,dimz);
  3: afisare_verificare(u,dimu)
  end;
  Readln
End.
```

În final propunem cititorului rezolvarea prin această metodă a următoarelor probleme.

Problema 3:

Se citește de la tastatură un număr natural k. *Să se determine un număr natural* $n \geq 3$ *și un șir* $x=(x(3),x(4),\ldots,x(n))$, $x(i)$ *din mulțimea* $\{-1,1\}$, $i \in \{3,4,\ldots,n\}$, *astfel încât*

$$k = x(3)\cdot C_3^3 + x(4)\cdot C_4^3 + \ldots + x(n)C_n^3$$, *unde* C_n^h

reprezintă numărul de combinări de n *luate câte* h *dintr-o mulțime cu* n *elemente.*

Exemplu:

Pentru k=7 o soluție este n=5 și x=(1,-1,1).

Indicație:

Se va folosi egalitatea

$C_{m+4}^3 - C_{m+3}^3 - C_{m+2}^3 + C_{m+1}^3 = 2m + 3$, pentru orice $m \geq 2$ număr natural.

Problema 4: *Se citește de la tastatură un număr întreg* k. *Să se determine un număr natural* $n \geq 3$ *și un șir* x=(x(4), x(5),...,x(n)),x(i) *dinmulțimea* {-1,1},i∈{4,...,n}, *astfel încât*

$$k = x(4) * C_4^4 + x(5) * C_5^4 + \ldots + x(n) * C_n^4$$

unde C_n^h *reprezintă numărul de combinări de* n *luate câte* h *dintr-o mulțime cu* n *elemente.*

Exemplu:

Pentru k=19 o soluție posibilă este n=6, x=(-1,1,1).

Indicație:

Se folosește relația:

$$C_{m+1}^4 - C_{m+2}^4 - C_{m+3}^4 + C_{m+4}^4 - C_{m+5}^4 + C_{m+6}^4 + C_{m+7}^4 -$$
$$- C_{m+8}^4 - C_{m+9}^4 + C_{m+10}^4 + C_{m+11}^4 - C_{m+12}^4 + C_{m+13}^4 -$$
$$- C_{m+14}^4 - C_{m+15}^4 + C_{m+16}^4 = 64$$

pentru orice $m \geq 3$ număr natural.

Problema 5: *Se citește de la tastatură un număr natural* k. *Se cere să se determine un număr natural* n *și un șir* x=(x(1), x(2),...,x(n)), x(i) *din mulțimea* {-1,1}, i∈{1,2,...,n}, *astfel încât* k=x(1)*13+x(2)*23+...+x(n)*n3.

Exemplu:

Pentru k=30 o soluție posibilă este n=4, x=(1,-1,-1,1).

Indicație:

Se folosește relația

(m+1)3-(m+2)3-(m+3)3+(m+4)3-(m+5)3+(m+6)3+(m+7)3-(m+8)3=-48,

pentru orice număr natural m.

Bibliografie:

1. Titu Andreescu, *O extindere a unei probleme de teoria numerelor*, Gazeta Matematică Nr. 1/1980;
2. Constantin Cocea, *Două teoreme de reprezentare de tip Erdos-Suranyi*, Revista de Matematică din Timișoara, Nr. 1/1998;
3. Ion Cucurezeanu, *Probleme de aritmetică și teoria numerelor*, Ed. Tehnică 1976;
4. W. Sierpinski, *Elementary Theory of Numbers*, Warszawa, 1987.

Doru Popescu Anastasiu, profesor, Liceul "Radu Greceanu", Slatina

Să fie oare adevărat?

UN algoritm de FACTORIZARE

de Constantin Săplăcan

Nota redacției: Am primit recent acest material, care ne-a uimit prin simplitatea soluției propuse. Dacă lucrurile stau așa, algoritmul RSA nu ar mai putea garanta o cifrare sigură, ceea ce ar bulversa o întreagă industrie. Nu am avut timp să verificăm corectitudinea algoritmului și nici să-i evaluăm complexitatea. O implementare ar impune dezvoltarea unei biblioteci pentru "aritmetică lungă", ceea ce nu este o sarcină banală. Așteptăm părerea voastră. (M.S.)

Algoritmii folosiți în criptografia cu chei publice se bazează pe faptul că deși este ușor să înmulțești două numere prime foarte mari este extrem de greu să factorizezi produsul lor. Acest produs este cunoscut ca *modulul public* sau cheia publică. A întâlnit în literatura de specialitate aprecieri de genul: "descompunerea modulului public N în P și Q atunci când N are circa două sute de cifre poate dura milioane de ani".

Vom prezenta în cele ce urmează un algoritm de factorizare a modulului public N în cele două numere relativ prime P și Q.

Algoritmul se bazează pe următoarele considerații geometrice:

a) Orice număr N poate fi reprezentat ca arie a unui dreptunghi având o latură egală cu unitatea, cealaltă latură având mărimea numărului N.

b) Problema reconstituirii numerelor prime din produsul lor este echivalentă cu următoarea problemă de geometrie: cunoscând doar aria unui dreptunghi putem să deducem mărimea laturilor sale? Dacă N este aria vom avea un singur dreptunghi ale cărui laturi să fie numerele întregi P și Q.

c) O teoremă din geometrie ne spune că locul geometric al vârfului unui dreptunghi care-și modifică lungimea laturilor, păstrându-și aria constantă, (atunci când două din laturile dreptunghiului sunt suprapuse pe axele de coordonate) este o ramură a unei hiperbole echilatere.

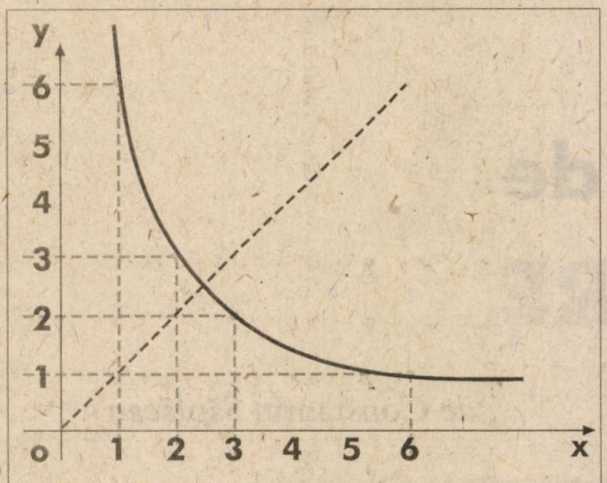
d) Hiperbola echilateră este simetrică față de prima bisectoare. Avem deci dreptunghiuri succesive de arie constantă. Doar lungimea laturilor se modifică: în timp ce mărimea uneia din laturi crește mărimea celeilalte scade. Aceste dreptunghiuri încep cu cel care are o latură egală cu unitatea și cealaltă latură de mărimea N și se termină în momentul în care vârful variabil al dreptunghiului se află pe prima bisectoare. Între aceste dreptunghiuri vom avea unul cu laturile de mărimea numerelor întregi P și Q.

Fiind vorba de numere întregi, vom trece de la considerațiile privind continuul geometric, la mărimi aritmetice discrete și vom reda ideile enunțate folosindu-ne de reprezentarea aritmo-geometrică a vechilor greci. Aceștia își reprezentau numerele ca mulțimi de puncte aliniate care, în funcție de dispunerea lor, permiteau să se vorbească de numere liniare, pătrate, dreptunghiulare. Un număr dreptunghiular va fi prin urmare un număr având m x n puncte dispuse pe m linii și n coloane. În cele ce urmează vom înțelege prin linii șirurile de puncte dispuse orizontal paralel cu axa absciselor Ox, prin "prima linie" linia cea mai apropiată de axă, prin "ultima linie" linia cea mai îndepărtată; prin coloane vom înțelege șirurile de puncte dispuse vertical paralel cu axa ordonatelor Oy, prin "prima coloană" cea mai apropiată coloană de axă, "ultima coloană" fiind cea mai îndepărtată coloană față de axă.

La fel cum orice număr poate fi reprezentat ca arie a unui dreptunghi având o latură egală cu unitatea, orice număr poate fi dispus sub formă liniară.

Dacă un număr are ca factori două numere întregi P și Q, acel număr poate fi reprezentat într-o manieră strict determinată ca număr dreptunghiular.

Inițial am găsit un algoritm de factorizare care consta într-un procedeu de trecere de la numărul dispus sub formă liniară la numărul dispus sub formă dreptunghiula-



Locul geometric al vârfului unui dreptunghi care-și modifică lungimea laturilor, păstrându-și aria constantă, este o ramură a unei hiperbole echilatere.

ră. Dar trecerea de la forma liniară la cea dreptunghiulară era extrem de lentă și atunci am căutat o metodă mult mai practică în care punctele sunt "aproape dispuse" în dreptunghiul căutat. Abreviat, algoritmul de factorizare poate fi prezentat astfel :

Din numărul N, modulul public, extragem radicalul. Cu partea întreagă a rădăcinii formăm un pătrat. Scădem din numărul N acest pătrat și diferența o dispunem pe coloană (sau coloane) în continuarea pătratului. Apoi începem "să coborâm" punctele de pe ultimele linii pe ultimele coloane pînă când obținem numărul dreptunghiular căutat.

Firește, nu vom lucra cu puncte efective, cu numere figurate, de acestea ne-am servit doar ca suport intuitiv al prezentării noastre. În realitate în loc de "mutări de puncte" vom face operații elementare cu numere: scăderi, adunări, împărțiri etc.

Din dorința de a fi bine înțeleși și din respect pentru cei care, asemenea lui Euclid, nu cunosc vremelnice limbaje de programare, vom prezenta algoritmul de factorizare în cel mai natural limbaj natural cu putință. Atunci când *există* un algoritm și este clar formulat în limbajul obișnuit, nu vor lipsi cei care să-l transpună într-un limbaj de programare.

Algoritmul de factorizare

Problema de rezolvat : Se cunoaște modulul public N. Se cere să se afle cele două numere P și Q al căror produs este N.

Vom avea:

Constante:
N := numărul total de puncte (modulul public)

Variabile:
Pl:= numărul de puncte de pe prima linie
Penl:= numărul de puncte de pe penultima linie

Criptografie cu chei publice

Bazele criptografiei cu chei publice au fost puse de Whitfield Diffie și Martin Hellman, de la Universitatea Stanford din California, printr-un articol celebru publicat în 1976. În locul unei singure chei secrete, ca în criptografia clasică, criptografia asimetrică folosește două chei diferite, una pentru cifrare, alta pentru descifrare. Deoarece este imposibilă deducerea unei chei din cealaltă, una dintre chei este făcută publică fiind pusă la dispoziția oricui dorește să transmită un mesaj cifrat. Doar destinatarul, care deține cea de-a doua cheie, poate descifra și utiliza mesajul.

Tehnica cheilor publice poate fi folosită și pentru autentificarea mesajelor, prin așa numită semnătură digitală, fapt care i-a sporit popularitatea. Folosind algoritmi cu cheie publică (asimetrici), se creează criptosisteme cu două chei, în cadrul cărora doi utilizatori (procese) pot comunica cunoscând fiecare doar cheia publică a celuilalt.

În criptosistemele cu chei publice, fiecare utilizator A deține o transformare de cifrare publică, E_A, care poate fi memorată într-un registru (fișier) public și o transformare de descifrare secretă, D_A, ce nu este posibil să fie obținută din E_A. Cheia de descifrare (secretă) este derivată din cheia de cifrare (publică) printr-o transformare greu inversabilă (*one-way*). În sistemele cu chei publice, protecția si autentificarea sunt realizate prin transformări distincte. Să presupunem că utilizatorul (procesul) A dorește să emită un mesaj, M, unui alt utilizator (proces) B. Dacă A cunoaște transformarea publică E_B, atunci A poate transmite M la B sub forma $C=E_B(M)$, asigurându-se astfel funcția de confidențialitate.

La recepție, B, va descifra criptograma C utilizând transformarea secretă DB, cunoscută doar de el:

$D_B(C)=D_B(E_B(M))=M$.

Cel mai cunoscut sistem cu chei publice este

Ul:= numărul de puncte de pe ultima linie
Pc:= numărul de puncte de pe prima coloană
Uc:= numărul de puncte de pe ultima coloană
Def:= "deficitul", respectiv numărul de puncte lipsă de pe ultima coloană pînă la nivelul "prima coloană minus 1"

Pasul 1. Extragem rădăcina pătrată din N. Fie a partea întreagă a acestei rădăcini.

Pasul 2. Formează pătratul a x a.

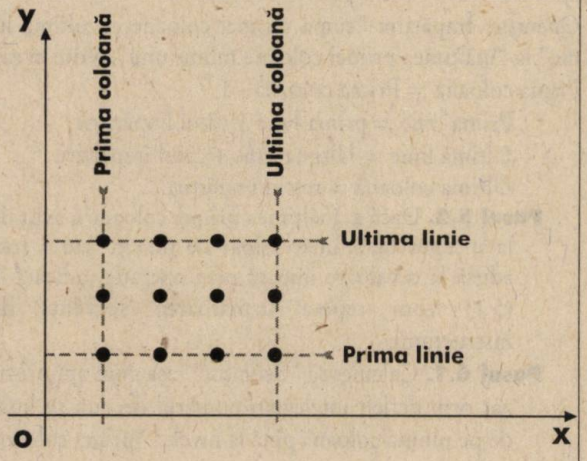
Pasul 3. Scade din N pătratul a x a. Rămîne un rest.

Pasul 4. Acest rest poate fi egal cu a, mai mare decât a sau mai mic decât a.

Pasul 4.1. Dacă restul scăderii este egal cu a atunci a e unul din divizorii lui N (ceea ce îndeobște nu e cazul).

Pasul 4.2. Dacă restul scăderii e mai mic decât a, se dispune acest rest pe o nouă "ultimă coloană". Vom avea:
Prima linie := a + 1
Prima coloană := a
Ultima coloană := restul scăderii
Ultima linie := a.

Pasul 4.3. Dacă restul scăderii e mai mare decât a, se împarte acest rest la a. În cazul în care a este unul din divizori, împărțirea se va face exact. Dacă împărțirea se face cu rest vom avea:
Prima coloană := a
Ultima linie := a + câtul împărțirii
Prima linie := a + câtul împărțirii + 1
Ultima coloană := restul împărțirii.



Un "număr dreptunghiular" este un număr având m x n puncte dispuse pe m linii și n coloane. În imagine, numărul 12 reprezentat prin 3 linii și 4 coloane.

Pasul 5. În cazul înălțimii primei coloane (respectiv a numărului a) putem întâlni două situații: când prima coloană are număr par de puncte sau când are un număr impar de puncte.

Pasul 5.1. În cazul în care înălțimea primei coloane este pară o vom aduce la o mărime impară. (Procedăm astfel deoarece numere P și Q nu pot fi numere pare).

Criptografie cu chei publice

RSA, al cărui nume provine de la de cei trei cercetători de la MIT (*Massachusetts Institute of Technology*) care l-au creat – Rivest, Shamir și Adleman. El este un adevărat standard „de facto" în domeniul semnăturilor digitale și al confidențialității cu chei publice. Se bucură de o foarte mare apreciere atât în mediul guvernamental cât si în cel comercial, fiind susținut prin lucrări și studii de comunitatea academică. Sub diferite forme de implementare, prin programe sau dispozitive hardware speciale, RSA este astăzi recunoscută ca cea mai sigură metodă de cifrare si autentificare disponibilă comercial. Publicat în 1978, RSA este bazat pe imposibilitatea practică, la nivelul performanțelor calculatoarelor de azi, de a factoriza numere prime mari. În același timp, găsirea unor numere prime mari este ușoară. Funcțiile de criptare/decriptare sunt exponențiale, unde exponentul este cheia și calculele se fac în inelul claselor de resturi modulo n.

Dacă p si q sunt numere prime foarte mari (100-200 de cifre zecimale), cifrarea și descifrarea se fac astfel:

$C=E(M)=M^e \bmod(p*q)$;

$M=D(C)=C^d \bmod(p*q)$;

Numerele e si d sunt cheile, publică și secretă. Valorile p și q sunt ținute secrete iar n=p*q este făcut public.

Cifrarea si descifrarea sunt bazate pe generalizarea lui Euler a teoremei lui Fermat, care afirmă că pentru orice M relativ prim cu n,

$M^{j(n)} \pmod n = 1$;

unde $j(n)$ este indicatorul lui Euler. Această proprietate implică faptul că e și d să satisfacă proprietatea:

$e*d \pmod{j(n)} = 1$,

Rezultă:

$M^{ed} = M \pmod n$.

Operație: Împărțim "suma ultimei coloane cu ultima linie" la "înălțimea primei coloane minus unu". Vom avea:
Prima coloană := Prima coloană - 1
Prima linie := prima linie + câtul împărțirii
Ultima linie := Ultima linie + câtul împărțirii
Ultima coloană := restul împărțirii.

Pasul 5.2. Dacă a, înălțimea primei coloane a avut de la început un număr impar de puncte sau a fost adusă la o valoare impară prin operația indicată la (5.1) vom repeta următoarea secvență de instrucțiuni:

Pasul 6.1. Calculează "deficitul" (așa cum am precizat prin deficit înțelegem numărul de puncte lipsă de pe ultima coloană pînă la nivelul "prima coloană - 1")
Deficitul := (Prima coloană -1) - Ultima coloană.

Pasul 6.2. Scade "deficitul" calculat din ultima linie și completează (adună-l la) ultima coloană.
Ultima linie := Ultima linie - Deficitul
Ultima coloană := Ultima coloană + Deficitul
Penultima linie := Prima linie (așa cum a fost calculată la 5.1)

Pasul 6.3. Adună punctele de pe penultima linie cu punctele de pe ultima linie și împarte suma rezultată la "numărul de puncte de pe prima coloană minus doi".

Se disting două cazuri: împărțirea se face cu rest sau fără rest. Când împărțirea se face cu rest avem
Prima coloană := Prima coloană - 2
Prima linie := Prima linie + câtul împărțirii + 1
Ultima coloană := restul împărțirii.

Pasul 6.4. Cât timp împărțirea se face cu rest repetă operațiile descrise la 6.1, 6.2, 6.3.

Când împărțirea se va face exact ultima coloană va fi unul din cei doi divizori căutați ai modulului public N.

Acesta este algoritmul de factorizare. Întrevedem două posibilități de îmbunătățire a lui, rezultate chiar din experiența elaborării. Așa cum am ținut cont de înălțimea pară a coloanelor am putea să ținem cont de lungimea pară a liniilor, numărul doi și deci numerele multiplii de doi nefiind printre numerele căutate. Cealaltă cale e încercarea de a trata în același mod coloanele care sunt multiplii de trei, (sau chiar de cinci, de șapte etc.). Și toate acestea astfel încît să nu ne complicăm mai mult încercând o simplificare.

Sperăm din tot sufletul ca acest algoritm să mai scadă din "milioanele de ani necesari factorizării unui modul public de două sute de cifre".

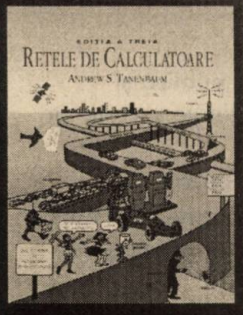
Iar în locul clasicelor "chei publice" vom propune un protocol de securitate a comunicațiilor inspirat din metoda de experimentare cunoscută în farmacologie sub numele "dublu orb".

Dl. Constantin Săplăcan din Cluj-Napoca poate fi contactat prin intermediul redacției. e-mail: office@ginfo.ro.

Prima carte românească de referință în domeniul rețelelor de calculatoare

Autor, profesor și cercetător, laureat al premiului ACM Karl V. Karlstrom Outstanding Educator Award, Andrew S. Tanenbaum, explică cu grijă felul în care rețelele funcționează în interior, pornind de la tehnologia hardware până la cele mai utilizate aplicații de rețea. Cartea urmează o abordare structurată a rețelelor de calculatoare, începând de jos (de la nivelul fizic) și parcurgând treptat calea spre vârful acestora (nivelul aplicațiilor).

Subiectele abordate includ:



- **Nivelul fizic (firele de cupru, fibrele optice, comunicațiile radio și cele prin satelit)**
- **Nivelul legătură de date (principiile protocoalelor, HDLC, SLIP și PPP)**
- **Subnivelul de acces la mediu (LAN-uri IEEE 802, punți, noile LAN-uri de mare viteză)**
- **Nivelul rețea (rutarea, controlul congestiei, interconectarea rețelelor, IPv6)**
- **Nivelul de transport (principiile protocoalelor de transport, TCP, performanța rețelei)**
- **Nivelul aplicație (criptografia, poștă electronică, știri, Web-ul, Java, multimedia)**

În fiecare capitol sunt descrise în detaliu principiile de bază, urmate apoi de exemple ample luate din Internet, rețele ATM și rețele de comunicații fără fir.

Doresc să cumpăr cartea Rețele de calculatoare

V-am expediat în contul C.P. Agora nr.: 2021400027100178 deschis la Banca Transilvania, filiala Mureș, 100.000 lei (în care nu sunt incluse taxele poștale), cu mandatul poștal/ordinul de plată nr.: ..... din data de ..... ☐ Doresc să-mi trimiteți și factură.

Pentru elevi și studenți cartea costă 75.000 lei. Ei vor trebui să atașeze comenzii o copie după carnetul de elev sau student.

Numele și prenumele: ..... Funcția: .....

Firma: ..... Adresa: .....

Cod poștal/Localitate: ..... Telefon/Fax: .....

Data ..... Semnătura .....

Se poate și fără recursie

Turnurile din HANOI în JAVA

de Florin Sasu

Problema turnurilor din Hanoi se pretează foarte bine la o abordare recursivă. În ciuda simplității expresiei, programele astfel realizate sunt mai puțin eficiente decât cele iterative, chiar dacă acestea din urmă sunt uneori mult mai complicate. Acest articol vă prezintă un astfel de exemplu.

În urmă cu vreo doi ani am citit și eu problema Turnurilor din Hanoi. Este o problemă deosebită.

Rezolvarea clasică pentru această problemă este cea recursivă. Fiind începător în informatică pe atunci, aveam despre algoritmii recursivi amintiri neplăcute (*Error 202*, *Stack Overflow Error*, sau chestii de genul acesta), așa că pur și simplu am căutat o rezolvare nerecursivă. Cea folosind Backtracking nu mi-a plăcut, fiind ineficientă; din două mutări face una "greșită" (adică nu se face numărul minim de mutări).

Numărul mic de tije cu discuri pare la început o piedică în rezolvarea problemei. Mi-am dat seama apoi că acest lucru este de fapt o limitare care duce până la urmă la o soluție unică. Observând cum "lucrează" algoritmul recursiv, am intuit că voi putea scrie un algoritm (nerecursiv) care să "știe" în fiecare moment ce mutare trebuie făcută.

Astfel am găsit o a treia condiție care, împreună cu cele două subînțelese din problemă, sunt suficiente pentru a descrie mișcarea celor n discuri de pe tija 'a' pe 'b'.

1. nu se mută disc mare peste disc mic;
2. nu se mută înapoi (dacă am mutat un disc k de pe 'a' pe 'c', nu-l voi muta apoi pe k de pe 'c' pe 'a'
3. nu se mută disc par peste disc par sau impar peste impar (considerând discurile numerotate după mărime)

Programul pentru această rezolvare funcționează teoretic pentru un număr foarte mare de discuri (l-am testat pentru 200 000). (Silly, isn't it ?)

Alături se află sursa în Java (tocmai învățasem Java când l-am scris). Numiți-l "Hanoi.java", compilați-l și apoi rulați-l cu numărul de discuri ca parametru. De exemplu: "java Hanoi 4" (4 - numărul de discuri).

```
import java.awt.*;
import java.io.IOException;

class Tija {
  int v[];
  int k;       //numărul curent de discuri pe tija
  Tija(){
    v=new int[20];
//max 20 de discuri, se poate modifica
  }
}

public class Hanoi {
  static int us,ud,n;
//us-ultima sursă, ud+ultima destinatie
  static Tija t[]=new Tija[3];

static boolean  //se încearcă mutarea unui disc de
incearca (int s,int d){ //pe tija s pe tija d
  if (t[s].k>0)
    if ((t[s].v[t[s].k]-
         t[d].v[t[d].k])%2!=0)
      if (t[d].k>0)
        if (t[s].v[t[s].k]<
            t[d].v[t[d].k])
          return true;
        else return false;
      else return true;
    else return false;
  else return false;
}

static void //se mută un disc de pe s pe d
schimb(int s,int d){
  System.out.println("se muta de pe tija "+
    (char)(s+97)+" discul "+t[s].v[t[s].k]+
    " pe tija "+(char)(d+97));

  t[d].k++;
  t[d].v[t[d].k]=
    t[s].v[t[s].k];
  t[s].k--;
```

```
  ud=d;
  us=s;
}

public static void
main (String args[]){
  if(args.length<1) {
    System.out.println(" un numar");
    try {
      System.in.read();
    } catch (IOException e) {
        return;
    }
    return;
  }
  String param=args[0];
  try{
    n=Integer.parseInt(param);
  }catch(NumberFormatException e)
  {System.out.println("un numar");
  return;
  }
//până aici am aflat numărul de discuri-n

  for(int i=0;i<3;i++)
    t[i]=new Tija();

  for(int i=0;i<=n;i++)
    t[0].v[i]=n-i+1;
    t[0].k=n;
  t[1].v[0]=n+1;
  t[1].k=0;
    t[2].v[0]=n;
    t[2].k=0;
  us=0;
  ud=(n%2)+1;
//initializări

  do{
    switch(us+ud){
    case 1:if (incearca(us,2)) schimb(us,2);
           else if (incearca(2,0)) schimb(2,0);
           else if (incearca(2,1)) schimb(2,1);
           break;
    case 2:if (incearca(us,1)) schimb(us,1);
           else if (incearca(1,0)) schimb(1,0);
           else if (incearca(1,2)) schimb(1,2);
           break;
    case 3:if (incearca(us,0)) schimb(us,0);
           else if (incearca(0,1)) schimb(0,1);
           else if (incearca(0,2)) schimb(0,2);
    } //switch
  }while(!(t[0].k==0 && t[2].k==0));
//rezolvarea propriu-zisa
    System.out.println("Apasati <enter>");
    try {
      System.in.read();
    } catch (IOException e) {
        return;
    }
  }//main
  }//Hanoi
  /*  javac Hanoi.java
      java  Hanoi 4
   */
```

Florin Sasu este elev în clasa a XII-a la Liceul "Spiru Haret" din București. Poate fi contactat prin e-mail la adresa drag@art.ro.

Recursiv

În LOGO:

```
to move :from :to
(print "Move "disk "on "peg :from "to "peg :to ".)
end

to tower :disks :from :to :using
if :disks = 1 [move :from :to stop]
tower :disks-1 :from :using :to
move :from :to
tower :disks-1 :using :to :from
end
```

În Pascal:

```
program tower;
procedure hanoi(nr: integer; from, onto, other:
char);
  procedure movedisk(nr: integer; from, onto: char);
    begin {movedisk}
      writeln('Move disk ',nr:1,' from ',from,
              ' to ',onto)
    end; {movedisk}

  begin {hanoi}
    if nr <> 0 then
      begin
        hanoi(nr-1, from, other, onto);
        movedisk(nr, from, onto);
        hanoi(nr-1, other, onto, from)
      end
  end; {hanoi}

begin {main program}
  hanoi(5, 'a', 'b', 'c')
end.
```

De ce Gates și nu Kildall?

Povestea DOS-ului

de Gabriel Proșcanu

Un episod straniu din istoria calculatoarelor: de ce a ales IBM tocmai pe Microsoft pentru dezvoltarea sistemului de operare pentru PC-uri?

Nici două decenii nu au trecut de la apariția primului IBM PC; cu toate acestea, întâmplările acelor zile rămân învăluite în acea aură mitică proprie începuturilor marilor aventuri.

De ce IBM a ales un procesor slab pentru PC-uri? De ce nu a folosit pentru acestea unul dintre sistemele de operare care erau deja binecunoscute la acea dată? De ce inventatorul DOS a rămas, până astăzi, anonim, iar Bill Gates a devenit celebru?

Participanții la acele evenimente ne oferă variante contradictorii asupra întâmplărilor din acele zile. Multe legende și multe distorsionări voite lasă nenumărate întrebări fără răspuns, dar, la fel ca în multe alte domenii, nu contează de unde s-a plecat, ci unde s-a ajuns...

De ce 8088 și nu 68000, care era un procesor mai puternic?

Răspunsul este simplu și, astăzi, ne face să zâmbim: deoarece IBM dorea ca PC-urile să fie niște mașini low-end, care să concureze cu mașinile CP/M și Apple II... nu, însă, și cu plănuitul "PC adevărat", care urma să apară, într-un scurt răstimp. Așa cum bănuiți, deja, "adevăratul PC" nu a mai văzut, vreodată, lumina zilei... Dar, pe vremea aceea, nimeni nu prevedea acest lucru. Primul PC a fost proiectat, de la început, ca fiind ruda săracă a mașinilor IBM; i-au acordat un procesor pe 16 biți, dar nu prea multă memorie. Ce dacă 8088 avea o magistrală pe 8 biți ? Era mai ieftin decât 68000 și asta conta în acel moment. 8088 adresa doar 1 MB de memorie (ca să nu mai vorbim, de faptul că proiectanții IBM i-au redus posibilitățile doar la 640 KB)? Păi cine ar putea avea nevoie de mai multă memorie pentru un PC ?

8088 avea avantajul portării ușoare a codului scris pentru 8080/Z80. Nu era un lucru de neglijat, deoarece asta însemna că extrem de multe programe puteau apărea foarte repede, inclusiv celebritățile acelor zile - Microsoft Basic și WordStar.

Folosim CP/M sau un alt sistem de operare?

Majoritatea cititorilor acestei reviste probabil că nu au văzut vreodată o mașină CP/M. CP/M începea să dispară pe vremea când ei se nășteau. De aceea, probabil că au doar o vagă idee despre Gary Kildall; nu voi intra în multe amănunte, dar cred că va fi suficient pentru caracterizarea personajului, să spun că el a fost întemeietorul firmei Digital Research și ocupa, în epocă, poziția pe care o are astăzi Bill Gates. Este de închipuit, prin urmare, că un asemenea personaj se credea și el centrul universului și își permitea niște gesturi incredibile - ca, de pildă, să-i facă pe capii IBM să-l aștepte în zadar cu un contract, în timp ce el se plimba cu avionul personal... Dar, să nu anticipăm...

Așa cum spuneam, CP/M era, în mod firesc, sistemul de operare pe care cei de la IBM preconizaseră a-l instala pe noile lor mașini. De aceea, un grup de personaje cu putere de decizie de la IBM, în frunte cu Jack Sams, au stabilit o întâlnire cu Kildall, pentru a perfecta afacerea. Întâlnirea nu a decurs așa cum ar fi trebuit, deoarece Gary avusese, deodată, chef să-și piloteze avionul personal și și-a trimis la acea întâlnire soția și avocatul. Nici cei de la IBM nu s-au lăsat mai prejos pentru a da un ton încordat discuțiilor, susținând, în ruptul capului, că soția lui Kildall și avocatul trebuie să semneze un acord de confidențialitate, foarte dezavantajos pentru Digital Research1. Cum ei nu au făcut acest lucru, Sams și însoțitorii săi s-au suit în avion și s-au dus la Seattle, unde au bătut palma cu Microsoft.

Acesta este versiunea celor de la IBM.

Kildall ne spune altă poveste. Da, era în avionul său personal, dar nu se plimba, ci zburase la San Francisco, pentru niște afaceri, iar la întâlnirea cu cei de la IBM ajunsese la timp. Nu numai atât, dar semnase și acel acord de confidențialitate și credea că afacerea era încheiată. Liniștit, plecase în Caraibe pentru o scurtă vacanță3, dar, la întoarcere, aflase, cu stupoare, că IBM semnase un contract cu Microsoft.

Ce povestește Bill Gates

IBM urmărea, de câtva timp, evoluția ascendentă a pieței computerelor personale. Până la începutul anilor '80, existaseră câteva încercări anemice de a-și promova produsele proprii, soldate cu eșecuri imediate, astfel că, la un moment dat, IBM luase în considerare și posibilitatea de a cumpăra compania Atari, a cărei linie de computere personale să o dezvolte. Președintele IBM din acea vreme a hotărât că alta era calea de urmat. A adunat în jurul său câțiva manageri de succes și a demarat un proiect cu numele de cod Manhattan, prin care se

urmărea crearea unui PC "conform cerințelor pieței". Cei de la IBM - zice Gates - au abordat Microsoft sub pretextul efectuării unui studiu al pieței, cerîndu-le să semneze un acord de confidențialitate (care ar fi permis IBM să susțină că întâlnirea nu a avut loc vreodată) și i-au cerut lui Bill părerea despre cum trebuie să arate un PC. "Nu am avut nici o problemă cu secretomania celor de la IBM", declară Gates, "iar despre cum ar trebui să arate un PC aveam o idee foarte clară". Acesta, conform spuselor sale, ar fi trebuit să se folosească de noul Intel 8086 pe 16 biți, pentru o performanță sporită și posibilități grafice superioare computerelor personale existente pe piață. IBM s-a arătat interesat de aceste idei și au revenit cu propunerea ca Microsoft să scrie o versiune Basic specială pe care să o includă în memoria ROM a noii linii de computere. Microsoft scrisese, deja, un Basic pentru 8086 și acordul a fost realizat imediat. "În mod firesc, noua generație de PC avea nevoie de un sistem de operare" spune Bill Gates, "așa că le-am spus celor de la IBM să ia legătura cu prietenul meu Gary Kildall, de la Digital Research, care scrisese CP/M, mai ales că Digital Research deja plănuia să scrie un nou sistem de operare, pentru 8086, numit CP/M 86".

Soarta...

Microsoft ținea morțiș să participe la acest proiect și s-au angajat să scrie un sistem de operare, deși experiența lor în domeniu era zero. Cu toate acestea, șansa avea să fie de partea lor, nu o dată, ci de două ori...

În primul rând, IBM avea o oarecare reținere să trateze cu o companie firavă, care producea un soft fără pretenții. Nici renumele de bun vânzător al propriilor produse, pe care Microsoft începea să și-l capete, nu a fost factorul determinant pentru IBM, ci un fapt incredibil: Mary Gates, mama lui Bill, lucrase în Consiliul Național al United Way, fiind colegă cu unul dintre președinții executivi de la IBM. Această referință - "băiatul lui Mary Gates" - a avut o pondere semnificativă în luarea deciziei de colaborare, dovedind, dacă mai era nevoie, valoarea ridicată a referințelor sociale.

Apoi, a fost șansa de a avea la îndemână un sistem de operare. După euforia semnării angajamentului, a urmat panica: cei de la Microsoft și-au dat seama că nu știau nimic despre cum trebuie scris un sistem de operare și dădeau din colț în colț... până când unul dintre ei și-a amintit că vorbise, acum câtva timp, cu un tip din Seattle4, care își construise singur un model de PC, în variantă proprie. Acela folosise procesorul 8086 și, neavând răbdare să aștepte până când Digital Research va scoate CP/M 86, își scrisese propriul sistem de operare, pe care îl numise QDOS - *quick and dirty operating system*...

Urmarea: Microsoft a cumpărat QDOS, l-a rebotezat MS-DOS, l-a ajustat pe ici, pe colo, apoi restul devine prea cunoscut pentru a mai fi povestit...

Note:

1 O mostră: în înțelegere se specifica faptul că DR nu putea dezvălui informații confidențiale către IBM, dar dacă totuși o făcea, IBM nu putea fi dat în judecată pentru folosirea acelor informații; în schimb, DR putea fi acționat în justiție, dacă ar fi folosit informațiile oferite de IBM;..

2 Kildall spune că povestea "plimbării cu avionul" a fost inventată de Gates, într-un interviu acordat lui London Times.

3 Ducând cu el, în avion, echipa de negociatori IBM, pe care i-a lăsat la Miami, după care el și-a continuat drumul.

4 Tim Patterson

Gabriel Proscanu este redactor permanent al revistelor PC Report și Byte România. E-mail: gproscanu@agora.ro

Garry Kildall

În buna tradiție a familiei, Gary Kildall ar fi trebuit să fie profesor la *The Kildall College of Nautical Knowledge* din Seattle, fondat de bunicul său în 1924. După terminarea liceului, Gary s-a dedicat studiului, încercând să uite o adolescență turbulentă și o evoluție școlară mediocră. S-a înscris la Universitatea din Washington și a studiat matematica. Pe parcursul facultății, însă, a început să se intereseze de computere, iar la terminarea facultății a început să studieze și CS (*computer science*). Încă bănuia că va fi profesor, de astă dată în domeniul computerelor. Totul părea să se îndrepte în această direcție, mai ales că avusese de ales între a merge pe front, în Vietnam, și a accepta un post de profesor la *The Naval Postgraduate School* din Monterey, California. Acolo a fost creat CP/M, care i-a adus faima și succesul, devenind sistemul de operare standard în lumea microcomputerelor. Ironia sorții e că Intel, căruia Kildall îi făcuse o ofertă, nu s-a arătat interesat de sistemul de operare, spunând că piața microcomputerelor este prea redusă pentru a obține vreun profit. Din această cauză, Gary și-a creat propria sa companie în 1974 (*"Microcomputer Applications Associates"*) și a început să-l distribuie pe cont propriu. La sugestia soției sale, compania s-a transformat în corporație: el programa, iar soția sa se ocupa de comercializare. Numele acestei corporații era *"Intergalactic Digital Research, Inc."*, care a devenit, puțin mai târziu, "Digital Research Inc.". Revoluția computerelor personale începea...

În 1977 Digital a semnat un contract cu IMSAI; acesta i-a permis distribuirea masivă a sistmului de operare, care s-a impus foarte repede ca un standard de necontestat. Toate celelalte companii din domeniul microcomputerelor au aderat la acest standard, care a avut o dominație fără egal, pînă în momentul în care IBM a introdus computerele personale.

IRAM sau memoria inteligentă

de Horia Giurgiu

Idee: În ciuda unei mari varietăți de memorii existente, se pare că performanța poate fi îmbunătățită semnificativ prin regândirea arhitecturii. Trebuie doar adăugat un procesor direct în capsula de memorie...

Atenția producătorilor de calculatoare a fost captată de un proiect de cercetare realizat de profesorul *Dave Patterson* de la departamentul de calculatoare al U. C. Berkeley.

La începutul anilor '80 acesta, împreună cu colaboratorii săi puneau bazele tehnologiei RISC (*Reduced Instruction Set Computing* — calcule folosind un set redus de instrucțiuni). Folosind în prelucrări doar cele mai frecvente instrucțiuni, s-a revoluționat proiectarea procesoarelor. Acest lucru a permis unor companii mici ca *Sun* și *SGI* să depășească substanțial produsele *Intel*. În 1987, Patterson a contribuit la dezvoltarea tehnologiei RAID (*Redundant Array of Inexpensive Disks* — matrice redundantă de discuri ieftine). Un astfel de sistem furnizează o capacitate de stocare mare și rapidă, la un preț scăzut, fapt care a permis ca astăzi aproape toate serverele mari de fișiere să folosească această soluție. Patterson începe un nou proiect intitulat IRAM (Inteligent RAM — RAM inteligent), care pornește de la ideea adăugării unui procesor în capsula de memorie. Aceasta ar conduce la o creștere semnificativă a performanțelor atinse de calculatoare.

Secretul succesului lui Patterson a fost înțelegerea profundă a factorilor economici care conduc industria de calculatoare. De exemplu cu RAID, el a remarcat că este mai rentabil să combine mai multe hard-disk-uri mici, decât să încerce să construiască unul singur de dimensiune mare.

Proiectul IRAM-ului este motivat de aceeași observație:

"Am scris un articol de sinteză pentru revista Scientific American și ceea ce mi-a atras atenția a fost faptul că avem două industrii producătoare de semiconductori: una pentru memorii și alta pentru procesoare" comenta Patterson. *"Deoarece costul liniilor de producție crește continuu, nu are sens să se construiască separat două fabrici, care sunt foarte asemănătoare"*.

Prin analiza posibilităților de combinare a memoriei și procesoarelor, el a realizat că IRAM-ul ar putea rezolva una dintre cele mai mari probleme ale proiectanților de calculatoare.

Problema a fost prezentată într-un mod dramatic în 1994 într-un manifest intitulat *"Ciocnirea de zidul memoriei: implicații ale evidentului"*. Lucrarea, scrisă de doi profesori de la Universitatea din Virginia, arăta — pe baza tendințelor curente relativ la viteza memoriei și a procesoarelor — că în mai puțin de o decadă performanța calculatoarelor va fi determinată

PRO-DC
STR. ARGEŞ NR. 21 - CLUJ-NAPOCA
286 1/20 monitor HGC mono $ 65
386SX 4/60 monitor VGA mono $ 150
486slc-75 8/260 monitor VGA color $ 295
486DX-66 8/270 monitor VGA color $ 330
preturile nu includ tva
garanție tehnică 12 luni
TEL-FAX 064-430.111
e-mail: prodtcom@mail.dntcj.ro

complet de timpul de acces la memorie. Din acel moment, oricât de mult ai îmbunătăți procesoarele, calculatoarele nu vor funcționa mai rapid.

Pentru a înțelege această problemă, să observăm evoluția lor. În ultimii 20 de ani, microprocesoarele și-au îmbunătățit viteza de sute de ori. Chip-urile de memorie și-au crescut capacitatea cu un factor de 256000. Dar viteza memoriei a crescut doar cu un factor de 10. Rezultatul? Procesoarele consumă tot mai mult timp așteptând date de la memorie, și din ce în ce mai puțin timp efectuând calcule utile.

Proiectanții de calculatoare au încercat în ultima vreme să amelioreze situația aceasta cu ajutorul memoriilor cache și a unor trucuri inteligente. Dar chiar și aceste tenhici sunt "covârșite" de distanța dintre viteza memoriei și a procesoarelor, care crește exponențial.

Motivul pentru care IRAM-ul ar putea să demoleze zidul memoriei este legat de modul în care sunt construite chip-urile de RAM obișnuite. Memoria principală a calculatorului este în general o variantă de RAM numită RAM dinamic (Dinamic RAM — DRAM). Un chip DRAM conține o matrice de celule de memorie, în care fiecare celulă conține un singur bit. Deoarece numărul de linii care pot intra în chip este limitat drastic, pentru a citi informația, prima dată se selectează rândul, apoi coloana, abia după care apare data cerută. Deoarece realizarea secvențială a acestor lucruri durează, o parte din întârzierea dată de memorie provine din timpul de selecție a biților doriți.

Pe lângă întârziere, proiectarea cu DRAM-ul obișnuit este paralizată de lățimea redusă a magistralei dintre procesor și memorie. Multe dintre chip-urile obișnuite de DRAM folosite în PC-uri au patru linii de ieșire. Aceasta înseamnă că putem citi doar patru biți consecutivi odată. Pentru a lăți acest "fir" subțire, mulți proiectanți de calculatoare leagă în paralel 8 chip-uri de DRAM pentru a ajunge la o magistrală de 32 de biți. Oricum, chiar și această creștere este tot mai scumpă și reprezintă încă o "strangulație" în calculator.

IRAM-ul rezolvă ambele probleme prezentate. Dacă prognozele se dovedesc corecte, s-ar putea reduce latența memoriei de două-trei ori, iar

Tipuri de memorii

SDRAM - Syncronous DRAM

Este o memorie performantă, care s-a impus deja cu succes în domeniul frecvențelor de magistrală de până la 100 MHz. Lucrează sincron cu procesorul, fără stări de așteptare suplimentare. Permite transferarea unei salve întregi de date (modul de lucru burst). Foarte important - rafalele de date citite pot fi programate - viteza și mărimea blocului. Organizarea sa pe două blocuri permite asigurarea unui flux neîntrerupt de date, trebuind doar accesate alternativ cele două bancuri independente. Datele sunt citite printr-un pipeline performant, care permite demararea următoarei cereri de acces înainte ca datele de la cererea curentă să fie deja valide. Integrat în echipamente video, SDRAM poate lucra la un debit de 135-200 MB/s, necesare rulării aplicațiilor video MPEG-II. Pentium II este dotat cu module DIMM cu cipuri SDRAM de 2 sau 4 tacți (acesta din urmă are viteza de răspuns mai ridicată).

SGRAM - Syncronous Graphics RAM

Este un SDRAM adaptat exigențelor foarte mari din echipamente high-end de grafică și este un competitor deosebit de puternic în domeniul graficii 3D. Permite citirea/înscrierea datelor în flux continuu, în blocuri mari, îmbunătățind astfel performanța controlerului. Datele sunt accesibile printr-un singur port, performanța sa foarte ridicată datorându-se arhitecturii specifice organizate pe două bancuri. Acesta permite accesarea simultană a două pagini diferite din memoria video.

EDO RAM

Memoriile EDO - Extended Data Output - sunt o variantă a tipului FPM. Au apărut odată cu procesorul Pentium și sunt destinate sistemelor rapide, începând cu Pentium 100 MHz. Comparativ cu memoriile clasice, viteza lor este cu cca. 30..40% mai mare, însă limita superioară a tactului magistralei este de doar 66 MHz, insuficientă procesoarelor moderne ale zilelor noastre. Recunoaște cu o mare precizie adresa cererilor următoare de date și le pregătește din timp, iar la terminarea accesului rămâne agățată de ultima adresă, accelerând foarte mult citirea datelor din vecinătate. Un avantaj major este disponibilitatea datelor la ieșire pentru mai mult timp (are la ieșire latch- uri), ele putând fi citite chiar si pe durata selectării adresei

lățimea magistralei ar putea fi crescută de 100 de ori. Reducea provine din simplul fapt că procesorul și memoria sunt unul lângă celălalt. Aceasta va elimina "timpul de acces" (timpul necesar semnalului să se propage între circuite) și va permite scheme de adresare mai sofisticate, adică mai rapide.

Creșterea spectaculoasă a lățimii magistalei provine din faptul că procesorul și memoria sunt pe același chip, ceea ce ne permite să realizăm nenumărate legături între ele. În schimb, realizarea de legături între circuite diferite este restrânsă sever de limitările date de consumul de energie, spațiu restrâns și costuri ridicate.

De altfel, IRAM-ul va îmbunătăți performanțele mai ales prin creșterea lățimii magistralei dintre memorie și procesor decât prin scăderea întârzierilor, adică exact ceea ce au nevoie o clasă de aplicații care se află în creștere acum. Aproape orice necesită grafică, precum compresia sau animația 3D, implică manipularea unor șiruri lungi de biți. IRAM deschide calea spre mult așteptata super-magistrală dintre procesor și memorie.

Cu toate acestea, IRAM-ul se confruntă cu obstacole importante. Poate cel mai important este reticența de-a dreptul istorică a producătorilor de DRAM de a se implica într-o soluție profitabilă nouă. Mai ales că, după ce ani de zile au făcut beneficii substanțiale din producția unor memorii proaste, de ce să schimbe acum?

Oricum, industria de DRAM se confruntă cu o criză dată de o cerere redusă și o capacitate de producție mare. Prin oferirea unei noi utilizări pentru liniile de fabricație, IRAM reprezintă o soluție de ieșire din această stare.

Poate că cel mai spectaculos lucru legat de IRAM este că netezește terenul de luptă. După cum arată *Michael Slanter*, editor la ***Microprocessor Report***, combatanți consacrați ca Intel nu sunt dornici să înceapă să construiască dispozitive de tip IRAM, deoarece nu știu nimic despre DRAM. Astfel că IRAM poate termina ceea ce Patterson a început cu RISC: *distrugerea supremației deținute de* ***Intel*** *și* ***Motorola***.

Dl. Horia Giurgiu este asistent la Facultatea de Calculatoare, Universitatea Tehnica Cluj-Napoca.

Tipuri de memorii

următoare (deși semnalul de selectare CAS trece pe nivel inactiv, datele de ieșire sunt disponibile în continuare, rezultând un efect pipelined pronunțat și ciclii mai rapizi). Viteza maximă este de 5-2-2-2, iar timpul de acces este de 50, 60 sau 70 ns (Atenție! doar cel de 60 ns poate lucra cu magistrale de 66 MHz).

BEDO RAM

Cerințele de acces pentru memoriile DRAM au crescut, si astfel a apărut memoria BEDO - Burst Extended Data Out RAM. Așa cum ne sugerează și denumirea, el lucrează prin bursting: sunt transferate la viteze mari blocuri mari de date. Acestea se constituie dintr-o multitudine de unități de date mai mici, care sunt transferate fără întrerupere, printr-o "salvă", de la adrese consecutive. Lucrează cu patru elemente de date în fiecare salvă: pregătește adresa de început, și când data este disponibilă, transferă și datele de la adresele următoare. Astfel ultimele trei elemente de date sunt obținute în timpi semnificativ mai mici, comparativ cu primul. Se descurcă doar cu magistrale de până la 66 MHz.

WRAM - Windows RAM

Prezintă foarte multe asemănări cu VRAM: este utilizat în exclusivitate în adaptoarele grafice performante, și are o arhitectură cu două porturi independente. Acesta permite reîmprospătarea imaginii pe ecran în mod continuu, chiar și în timp ce se scriu date în memoria video. Avantajul său major însă este viteza de transfer foarte ridicată, în medie cu 25..50%, datorită tamponului intern de memorie îmbunătățit. Practic WRAM permite rate de împrospătare îmbunătățite față de soluțiile bazate pe VRAM la prețuri de cost cu 20 % mai mici.

Video RAM

Este memoria folosită de plăcile video performante. Este adresabilă separat atât pentru scriere cât și pentru citire, prin două de porturi separate. Astfel, în timp ce procesorul grafic citește datele conținute, microprocesorul PC- ului poate înscrie noile date. Memoriile DRAM clasice nu permit accesul simultan a celor două procesoare pentru operații de I/O, și astfel refresh- ul imaginii era mereu întreruptă de înscrierea noilor date.

Incitant, terifiant...

MILLENNIUM Bug

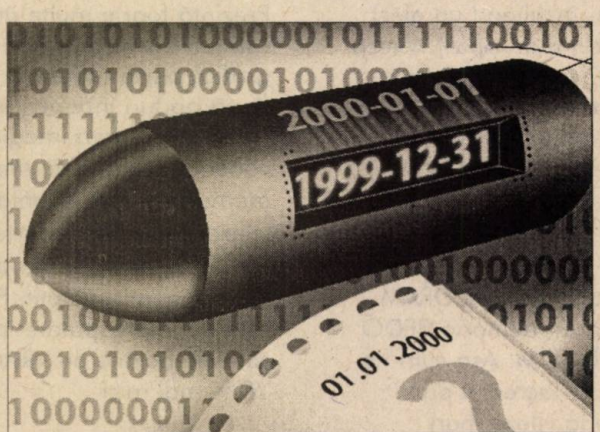
De Mircea Sârbu

Secole de-a rândul, astrologi, cabaliști și profeți de ocazie au legat sfârșitul mileniului al doilea de "sfârșitul lumii". Astăzi toate aceste preziceri stârnesc, cel mult, zâmbete. Când, la începutul anilor '90, unii specialiști în informatică au avertizat că foarte multe aplicații vitale sunt pândite de un iminent dezastru la cumpăna mileniului, prea puțini le-au dat crezare. Acum, când ne mai despart mai puțin de 700 de zile de anul 2000, lumea întreagă aude ticăitul unei imense bombe cu ceas...

Nu cu mult timp în urmă, pe adresa domnului Merlin, moderatorului forumului "PC Concrete", a sosit o întrebare în aparență banală:

"Am auzit în ultimul timp despre criza anului 2000. Mai toate calculatoarele se vor bloca atunci, pentru că nu vor mai putea schimba data. Există vreo soluție accesibilă mie, cel care lucrează pe un computer mai vechi (486)? Poate un program în Pascal sau C++ ar ajunge."

Am realizat atunci că "problema mileniului" (sau "criza anului 2000", sau "Y2K") este foarte adesea greșit înțeleasă. Mai cu seamă tinerii, care au "intrat" în informatica în epoca PC-urilor, nu au cum să înțeleagă un fenomen care se datorează unor stări de lucruri de acum mai bine de 20 de ani. În plus, chestiunea în cauză aduce în discuție un aspect foarte important al programării: tipurile de date.



Scurt și direct

"Criza anului 2000" nu este determinată de "data sistemului". Cu toate acestea, e bine să faceți un experiment: setați data la 31 decembrie 1999 și ora la 23:59. Opriți calculatorul și porniți-l după un minut. S-ar putea să aveți surpriza să constatați că ați călătorit în timp și că data curentă este 1 ianuarie 1980 sau 1984 (în funcție de BIOS-ul folosit). Dar, să admitem că toate calculatoarele vor putea schimba data și nu se vor bloca (sau, cel puțin, nu din această cauză).

Problema este de altă natură: Foarte multe aplicații economice dezvoltate în anii '60 - '70 se mai afla încă în funcțiune. Aceste aplicații au fost în mare parte dezvoltate în limbajul COBOL și rulează pe calculatoare mainframe sau mini (ele sunt așa-zisele "aplicații vitale moștenite"). Fiindcă la vremea când au fost dezvoltate memoria era o resursă foarte scumpă, proiectanții au "economisit" doi octeți din reprezentarea anului în datele calendaristice, fără să-și imagineze că aceste aplicații vor supraviețui mai mult de un sfert de secol...

Mai mult, această modalitate de reprezentare, fiind foarte larg utilizată la vremea respectivă, s-a impus ca un standard *de facto*, fiind acceptată și utilizată chiar și atunci când costurile stocării au devenit accesibile iar "economia de doi octeți", derizorie.

Desigur, lucrurile nu par foarte încă foarte grave, mai ales la noi, unde sistemele mainframe și mini în funcțiune sunt foarte puține... Dar în Statele Unite și unele țări occidentale există încă destul de multe astfel de calculatoare. Ceea ce este mai grav este faptul că aplicațiile moștenite care se rulează și acum sunt, de regulă, extrem de importante și de complexe. Acesta este și motivul pentru care nu au fost înlocuite. Există companii care n-ar putea supraviețui nici măcar câteva zile fără aceste sisteme.

Dar mai e o problemă: multe aplicații vechi, moștenite, au fost "modernizate" și transpuse pe rețele de PC-uri. Uneori au fost înfrumusețate cu interfețe utilizator prietenoase, însă - în esență - ele au păstrat structurile de date și procesările importante din vechile aplicații (care uneori au fost transpuse automat). Pentru multe programe este greu de știut de când datează de fapt anumite rutine. Aceasta este o problema care poate să creeze "surprize" și pe la noi.

Consecințe

Foarte pe scurt, problema mileniului implică două sarcini:

"În 1990, în Danemarca, o femeie în vârstă de 107 a primit o scrisoare de la inspectoratul școlar local prin care era invitată în clasa I-a. Un bărbat de 103 ani a fost internat într-un spital din SUA, unde a fost automat trimis la secția pediatrică. Toate acestea sunt fleacuri în comparație cu cele ce ne așteptă în anul 2000, dacă fiecare organizație care folosește calculatoare nu începe încă de pe acum să trateze problemele antrenate de faptul că 00 este mai mic decât 99."

(Case Strategies, Dec. 1992)

(1) verificarea și refacerea (dacă este cazul) tuturor programelor, astfel încât să folosească o reprezentare a anului din 4 cifre;

(2) convertirea tuturor fișierelor de date în conformitate cu această reprezentare. În cazul în care aceste conversii nu vor fi implementate, este posibil să apară anomalii după anul 2000.

Cam ce ar putea însemna aceste anomalii? Este gestul de greu de prevăzut. Scenariul alarmiștilor este o adevărată apocalipsă birocratică: morții se vor amesteca cu cei vii, bebelușii vor primi prin poștă pensia de bătrânețe iar moșnegi demult senili vor fi chemați la oaste; cetățeni onorabili vor fi somați să plătească milioane de dolari penalizări pentru întârzierea achitării facturilor iar norocoșii vor ridica dobânzi pe o sută de ani pentru sume depuse în bănci cu câteva zile mai devreme...

Mulți dintre proiectanții de atunci s-au pensionat, unii au murit... Adesea sursele programelor nu mai sunt disponibile sau, dacă sunt, foarte puțini le mai înțeleg. Nu-i de mirare că în Statele Unite programatorii în COBOL, câți au mai rămas, sunt la mare preț. Diversitatea reprezentărilor, codificărilor, limbajelor de programare și sistemelor de operare folosite fac ca nici un sistem automat să nu aibă șanse reale să rezolve problema *în mod sigur*.

Ar mai fi de menționat un aspect interesant: mulți analiști consideră că cei cărora li se vor bloca sistemele pe data de 4 ianuarie 2000 se vor putea considera... norocoși! Mult mai grave vor fi problemele generate de aplicațiile care vor continua să funcționeze. Datorită schimburilor generalizate de date, este posibil ca unele erori să se propage prin zeci de aplicații, alterând date din foarte multe sisteme.

Socotelile estimative consideră că "economia de doi octeți" va costa acum *cel puțin* 50 de miliarde de dolari.

Istoria

Cum s-a ajuns la această situație? Sunt multe aspecte care trebuie luate în considerare și nu toate sunt de ordin tehnic.

În primul rând trebuie să remarcăm că nu există practic o modalitate unanim acceptată de exprimare a datei, chiar și în afara calculatorului. Pentru a nu mai reface întreaga istorie a diverselor calendare până la cel gregorian actual, e suficient să amintesc că o notație de genul "2/1/98" va fi interpretată de un european ca 2 ianuarie 1998, în timp ce un american va înțelege 1 februarie 1998.

Cu atât mai puțin se poate vorbi de o reprezentare digitală standard a datei calendaristice. Deși nu este strict o problemă de limbaj de programare, voi menționa că în limbajul COBOL, o variabilă care reprezenta data calendaristică era declarată de obicei cam în felul următor:

```
01 DATA.
02 AA PIC 99.
02 LL PIC 99.
02 ZZ PIC 99.
```

De fapt, este o structură de tip articol *oarecum* echivalentă cu ceea ce în Pascal s-ar scrie:

```
type data_cal = record aa, ll,
zz string[2] end;
var data: data_cal;
```

Ciudățenia este că se folosea `string[2]` și nu `integer` (în COBOL, `COMP`), care ar fi fost mai eficient: ar fi permis reprezentarea pe doi octeți a anului (`aa`) până în 32.768, ceea ce cred că ar fi fost suficient... Ca să nu mai vorbim că, de fapt, `string[2]` se reprezintă intern pe 3 octeți.

Totuși, programatorii de atunci au preferat această formă, deoarece câmpurile astfel descrise foloseau intern reprezentarea așa-zis "zecimal-despachetată", ceea ce însemna că cifrele erau reprezentate prin caractere (ASCII sau EBCIDIC), dar - spre deosebire de structura Pascal descrisă - câmpurile erau considerate numerice și admiteau toate operațiile aritmetice. Avantajul era că, atunci când erau scrise într-un fișier, datele puteau fi citite cu orice editor de text - ceea ce era foarte important în depanare și întreținere.

De ce anul înaintea lunii și a zilei? Simplu: pentru că relația de ordine

lexicografică aplicată pe cele 6 cifre coincide cu ordinea temporală. Adică '980423' este mai mare decât '971230', ceea ce este corect (23 aprilie 1998 este *după* 30 decembrie 1997). Aceasta facilita comparațiile și permitea sortarea simplă a fișierelor prin utilitare de sistem. De asemenea, această relație de ordine a fost și este folosită în fișierele de index.

Din nefericire, această codificare pe doi octeți permitea doar reprezentarea numerelor între 0 și 99. Dacă depășirea zecimală nu e tratată în program, atunci adunând o unitate la 99 se obține 0. Și de aici încep necazurile... În general, programele consideră implicit că primele două cifre ale anului sunt 19, astfel încât după 1999 urmează 1900, cu tot cortegiul de consecințe.

Au fost însă și programatori care au tratat problema într-o manieră mai elegantă, utilizând o "fantă de timp" de 100 de ani. Stabilindu-și un reper (de pildă 70), considerau că tot ce este peste 70 are în față 19 iar ceea ce este mai mic sau egal cu 70 are în față 20 acoperind astfel intervalul cuprins între 1971 și 2070. Dar, în acest caz, relația de ordine trebuia implementată în program, deoarece nu mai coincidea cu cea lexicografică... Pe de altă parte, în cele mai multe aplicații, mai multe programe acționează asupra unui set comun de date. Cum aceste programe au fost scrise în perioade diferite de timp, de diverși programatori, este foarte greu de aflat ce convenții s-au folosit. Este posibil ca implementările diferite să fie compatibile doar până al 31 decembrie 1999...

Și încă ceva...

De parcă n-ar fi de ajuns, mai apar și alte câteva "mici" probleme.

În ce zi a săptămânii va cădea 1 ianuarie 2000? Orice calendar ne spune că va fi sâmbăta. Dar algoritmii care se bazează pe ultimele două cifre ale anului vor declara că e luni, pentru că așa a fost în 1900. Greșeala se va propaga în toate zilele următoare...

Spre deosebire de anul 1900, anul 2000 este bisect. Dacă programele nu au implementat complet algoritmul de detectare a anilor bisecți, cu siguranță ziua de 29 februarie va fi "sărită"...

În fine, mulți programatori au avut nevoie de niște date calendaristice "speciale" și - în absența unor standarde - au definit niște reguli cu valabilitate limitată la anumite aplicații. De pildă, pentru termene de genul "cât mai repede cu putință" au folosit date de genul "000101", iar pentru "cât mai târziu" ceva de genul "991231" (31 decembrie 1999). Aceste date "speciale" erau fie tratate în programele de aplicație, fie erau folosite doar pentru ca sortările și indexările să le aducă plaseze într-o anumită extremitate (în față sau în spate). Toate aceste date "speciale" vor interfera acum cu datele reale și este imposibil de prezis care vor fi consecințele pentru fiecare caz în parte.

Istoria timpului

Orice dicționar definește calendarul ca fiind un sistem de divizare a timpului bazat pe fenomene astronomice ciclice. Astfel, ziua este durata unei rotații a Pământului în jurul axei sale, luna este durata de rotație a Lunii în jurul Pământului iar anul este durata de rotație a Pământului în jurul Soarelui. Este interesant de remarcat că săptămâna nu corespunde unui fenomen astronomic. În plus, durata de rotație a lunii nu este fixă, media anuala fiind de cca. 29,5 zile. Cu toate acestea, aproape toate civilizațiile antice au folosit calendare lunare – excepția notabilă fiind calendarul solar al vechilor egipteni.

Calendarul iulian

Calendarul utilizat de oameni (pe care-l voi numi în continuare "civil") este doar o aproximare a "calendarului astral". Inițial importanța acestuia era preponderent de ordin religios dar, odată cu nașterea marilor imperii, importanța birocratică a calendarului a devenit prevalentă. Astfel, rigoarea organizatorică a Imperiului Roman în perioada sa vârf a venit în conflict cu aproximările prea grosiere folosite până atunci (bazate pe ciclurile lunare). Echinoxul de primăvară (fenomen astronomic care nu implică Luna) ajunsese să fie decalat cu trei luni, ceea ce crea serioase probleme birocratice. Împăratul Iulius Cezar l-a însărcinat în anul 46 Î.H. pe astronomul grec Sosigene să elaboreze un calendar mai precis și mai stabil.

Calendarul solar elaborat de Sosigene este prototipul celui modern. El se bazează pe faptul că anul astral are de fapt mai mult de 365 de zile (este mai lung cu cca. 6 ore). Anul 45 Î.H. a avut 445 de zile – pentru a corecta erorile acumulate – și s-a stabilit ca la fiecare patru ani (ani bisecți) să se adauge o zi suplimentară în luna februarie.

Calendarul impus de Iulius Cezar (numit "calendar iulian") a reprezentat un mare pas înainte, dar nu s-a dovedit totuși suficient de precis. De fapt anul tropic durează 365 de zile, 5 ore, 48 de minute și 36 de secunde. Diferența de 11 minute și 24 de secunde se acumulează în timp, creând ani bisecți în plus la fiecare 385 de ani. Astfel încât în anul

Să trecem peste faptul că cele două cifre ale anului au fost adesea folosite ca parametrii în generatoare de hazard (uneori pe post de împărțitor!), în scheme de serializare sau în mărci de timp (*timestamps*)... Și nu doar programele de aplicație dezvoltate *in-house*. Multe programe comerciale larg folosite (de pildă cele de calcul tabelar) ratează anumite manevre pentru date din anul 2000.

Profilaxie

Ce se poate face acum? E o problemă asupra căreia se concentrează atenția a mii și mii de specialiști. Există mai multe posibile soluții, unele temporare, altele permanente, dar fiecare are costurile ei și nici una nu este general valabilă.

Cum medicii spun că o boala este mai ușor de prevenit decât de vindecat, în cele ce urmează vă propun să ne concentrăm asupra unor măsuri de profilaxie. Nu am speranța că aplicațiile pe care le scriem noi vor supraviețui până în anul 9999, dar lecția pe care o putem trage din "scama mileniului" (*millennium bug*) este valabilă în foarte multe situații pe care nu le putem prevedea... În plus, atenție: la fiecare 3300 de ani apare un decalaj de o zi între calendarul astral și cel gregorian (vezi caseta "Istoria timpului").

Deoarece am petrecut câțiva ani din viață ca programator profesionist și am acumulat o oarecare experiență, îmi permit să vă ofer câteva sfaturi, pe care le voi grupa în două categorii: formale și tehnice. Cele tehnice vor porni de la noțiunea de "tip de dată" și vor propune trei posibilități de abordare.

Formalități...

1. Scrieți cod "curat". O mare parte dintre problemele pe care le pune repararea vechilor programe pentru a evita "scama mileniului" se datorează faptului că programatorii au scris un cod foarte alambicat și insuficient structurat (numit în jargon "*spaghetti code*"). Ați încercat vreodată să citiți codul scris de un coleg? Cu siguranță ați remarcat că este, în general, mult mai greu de înțeles codul propriu. În primii ani de meserie, am avut adesea probleme în descifrarea propriilor mele programe.

Gândiți-vă că un program are un ciclu de viață. Condițiile se schimbă, ansamblul aplicației se schimbă și el, astfel că - mai devreme sau mai târziu - orice program va ajunge din nou "sub bisturiu". Un cod clar, denumiri sugestive de variabile, o structurare perfectă, toate acestea vor economisi zile întregi de muncă și nervi.

Feriți-vă de "șmecherii" și "trucuri", chiar dacă ele par să sporească eficiența programului. Pe de o parte, cele mai multe programe au anumite zone "de gâtuire" (pe care le puteți depista cu un *profiler*), așa încât este foarte probabil că cele mai savante optimizări pe care le faceți să nu aibă nici un efect sau să aducă performanțe

---

1582 diferența ajunsese de zece zile.

Calendarul gregorian

Probabil că lucrurile ar fi mers în continuare așa, dar a intervenit o problemă. Lumea creștină celebrează Sfintele Paști conform unui calendar mixt (solar și lunar). Prin hotărârea conciliului de la Miceea din anul 325, sărbătoarea este stabilită în prima duminică *după* ziua cu lună plină care coincide cu echinoxul de primăvară (în mod normal 21 martie) sau care urmează imediat după acesta. Pentru a reglementa problema și a remedia lucrurile, Papa Grigore al XIII-lea l-a însărcinat pe astronomul Clavius să elaboreze un nou calendar.

Astfel ia naștere calendarul gregorian, folosit și astăzi. Pentru a elimina decalajul, după data de joi, 4 octombrie, a urmat vineri 15 octombrie. Pentru a împiedica acumularea erorilor de aproximare, calculul anilor bisecți a fost extins: dintre anii divizibili cu patru se exclud anii divizibili cu 100, exceptându-i pe cei care sunt divizibili cu 400.

Fiind elaborat de Biserica Catolică, țările preponderent protestante și ortodoxe au avut rețineri privind adoptarea calendarului gregorian. Germania l-a adoptat abia în 1700, Anglia și coloniile ei din America în 1752, Rusia în 1918, România în 1919 iar Grecia în 1923. Să remarcăm că în acest timp decalajul a crescut la 11 zile și că din istoria tuturor țărilor care au trecut de la calendarul iulian la cel gregorian "lipsesc" 10 sau 11 zile. Unele comunități ortodoxe nu recunosc nici astăzi calendarul gregorian. Au intervenit diverse probleme legate de desemnarea datelor ("pe stil vechi" și "pe stil nou"). Astfel, sovietici serbau revoluția bolșevică "din Octombrie" în... 7 noiembrie!

Unii nu vor...

Calendarul gregorian este astăzi acceptat ca referință în lumea întreagă, deși unele țări utilizează calendare proprii. Să amintim cu acest prilej că Revoluția Franceză a instituit printr-un decret din 24 octombrie 1793 un așa-zis "calendar republican", care a rămas în vigoare până la 1 ianuarie 1806. Anul "republican" era compus din 12 luni a câte 30 de zile (cu denumiri diferite de cele utilizate actualmente). La acestea se adăugau 5 sau 6 zile "complementare" dedicate sărbătorilor. Fiecare lună era împărțită în 3 decade a câte 10 zile.

Oare cum spuneau francezii de atunci *"I Hate Mondays"*?

suplimentare nesemnificative. Pe de altă parte, puterea calculatoarelor crește în mod constant (procesoare, memorie, disc etc.) astfel încât, oricum, eficiența programului va crește în timp. Dacă, totuși, recurgeți la anumite trucuri, încercați să le izolați cât mai bine în cadrul programului și să le comentați cât mai complet.

Nu pierdeți din vedere portabilitatea. În acest sens, este indicat să nu utilizați în programe elemente specifice sistemului de operare, procesorului și perifericelor cu care lucrați. Folosiți constante simbolice. Gândiți-vă ca este posibil ca programul să fie prelucrat în viitor de alte programe, de pildă pentru a fi adaptat unui alt calculator sau pentru a putea beneficia de procesarea distribuită.

2. Documentați programele. Celor mai mulți programatori nu le place să scrie documentații și - recunosc cu rușine - am făcut și eu parte din această majoritate. Dar timpul aparent pierdut cu documentarea programului este recuperat din plin în munca de întreținere și exploatare. În general, fiecare producător de software sau fiecare întreprindere dispune de o metodologie proprie, care consemnează convențiile utilizate în documentații. Veți fi obligați să le respectați. Această metodologie va stabili cu siguranță o modalitate unitară de numerotare a versiunilor precum și una pentru datarea modificărilor, o codificare a mesajelor de eroare etc. Există însă și o mare marjă de libertate, pe care vă recomand să o folosiți mai cu seamă în două scopuri.

În primul rând, documentați foarte clar interfețele dintre diferite componente ale aplicației. Acestea trebuie să reprezinte adevărate contracte, respectate pe toata durata de viață a aplicației. Păstrând stabile aceste interfețe, veți putea schimba implementarea anumitor rutine fără ca aceste modificări să afecteze în mod imprevizibil funcționarea aplicației în ansamblu.

În al doilea rând, notați cu precizie relațiile dintre diferitele rutine, astfel încât să puteți afla imediat ce rutine sînt apelate dintr-o anumită rutină și care sunt rutinele care apelează o rutină dată. Există - sau se pot scrie - programe care să analizeze codul sursă și să depisteze automat aceste relații.

3. Programați "defensiv". Această tehnică nu este definită riguros nicăieri și constă, de fapt, dintr-un set de reguli de bun simț. Iată doar câteva, pe care le puteți utiliza încă de mâine:

Este preferabil ca un program să se oprească și să dea un mesaj de eroare inteligibil, decât să furnizeze rezultate eronate.

Verificați toate informațiile furnizate de utilizator. Este om și poate greși. Validările trebuie să semnaleze valorile "dubioase" și să ceară o confirmare specială.

Tratați în program toate cazurile extreme, indiferent cât de improbabile par. Începând de la absența unui fișier, citirea unui fișier vid sau lipsa de spațiu pe disc și până la evenimente externe (de pildă căderi de tensiune), totul trebuie prevăzut.

Utilizați toate facilitățile limbajului pentru tratarea erorilor. Directivele ASSERT și mecanismele de tratare a excepțiilor (instrucțiuni `throw-try-catch` în C++ și Java) sunt primele pe care trebuie să le aveți în vedere. Daca limbajul nu vă ajută, implementați mecanisme proprii (fluxuri de mesaje, valori speciale returnate de funcții etc.).

Feriți-vă de variabilele globale! Întregul flux de informații din program trebuie să treacă prin interfețe, altfel așa-numitele "efecte laterale" vor face incontrolabilă orice aplicație de dimensiuni mari.

Tipuri de date

Aspectul tehnic fundamental pe care îl relevă "scama mileniului" este, de fapt, problema tipurilor de date. Așa cum s-a arătat în articolul "Tipuri de date" de Mihai Budiu (GInfo 1/1998), un tip de date nu reprezintă doar o mulțime de valori, ci și definiția completă a setului de operatori care pot acționa asupra acestor valori. În cele ce urmează voi utiliza o terminologie diferită: *domeniul* tipului (mulțimea valorilor) și *semantica* tipului (operatorii aplicabili).

Ca și bătrânul COBOL, limbajul Pascal standard furnizează un set limitat de tipuri, numite fundamentale, pe care compilatoarele de regulă le implementează. Spun "de regulă" pentru că este posibil ca anumite compilatoare să implementeze alt set de tipuri sau să implementeze diferit unele tipuri (de pildă, compilatoarele Pascal realizate de Borland implementează altfel șirurile de caractere decât o cere standardul oficial al limbajului). Referindu-ne în continuare doar la Pascal, știm că programatorul poate să definească noi tipuri de date prin derivare, pornind de la cele fundamentale. Este însă important să remarcăm că derivarea noilor tipuri se limitează la domeniul tipului. Semantica este fie moștenită de la tipurile de bază (ca în cazul subdomeniilor), fie este predefinită (ca în cazul enumerărilor și a tipurilor structurate).

Nici limbajul COBOL și nici limbajul Pascal nu furnizează un tip predefinit care să implementeze semantica datei calendaristice. Bazele de date relaționale au început să ofere un astfel de tip abia la sfârșitul anilor '80, iar standardul SQL l-a consemnat mult mai târziu. Programatorii nu au putut decât să improvizeze, astfel încât adesea semantica datei calendaristice a fost implementată empiric, în amestec cu logica aplicației.

Care ar fi variantele mai bune? Actualmente, limbajele orientate pe obiecte permit definirea completă a tipurilor utilizator, iar unele sisteme permit construirea de "componente software" care să cuprindă tipuri de date definite de utilizator.

Cum am putea totuși să implementăm într-un mod elegant (fără a face, încă, apel la tehnicile *object-oriented*) un tip de dată care să beneficieze de semantica datei calendaristice "grégoriene"? Principiile de bază ale ingineriei programării ne recomandă "încapsularea" acestei implementări, așa încât varianta directă este crearea unei biblioteci de

rutine care să ofere toate operațiile utile.

Deoarece crearea bibliotecilor "dinamice" este foarte dependentă de sistemul de sistemul de operare (în Windows există așa-numitele DLL-uri - *Dynamic Link Library*) vom recurge la o bibliotecă statică, concretizată într-un unit Pascal.

Unit-ul CALENDAR

În listingul CALENDAR.PAS am schițat o posibilă interfață pentru un modul (unit) care să trateze operațiile legate de date calendaristice. Nu pretind că este cea mai bună variantă posibilă, însă cred că poate fi utilă ca "îndrumar" în realizarea unei implementări coerente. Să discutăm câteva aspecte.

În primul rând, decizia ca reprezentarea publică a tipului `Data` (dată calendaristică) să fie un șir de caractere este, desigur, discutabilă. Argumentul cel mai puternic pentru această formă este că utilizatorii acestui tip au la dispoziție operatorii impliciți de comparare a stringurilor (așa cum am remarcat, reprezentare *an-lună-zi* face ca ordinea lexicografică să coincidă cu ordinea temporală). Aceasta facilitează și utilizarea unor algoritmi de sortare sau de indexare.

Alternative posibile ar fi reprezentările binare pe care le-am sugerat pentru implementare:

```
Type
        Normal = record
            Y, M, D: Word;
            end;
        Julian = record
            Y, D: Word
            end;
```

Al doilea tip (`Julian`) este numit "reprezentarea iuliană" (nu are nimic de-a face cu calendarul iulian), `D` fiind numărul de zile de la începutul anului. De exemplu, pentru 13 martie 1998 am avea Y=1998 și D=72 (însă într-un an bisect D=73).

Există și varianta de a reprezenta datele prin numărul de zile trecute de la prima zi a anului 1. De pildă, 1 ianuarie 1998 s-ar reprezenta prin 729392.

Când începe mileniul III?

Stabilirea unor repere temporale solide a dat de furcă istoricilor. Principalele probleme sunt legate de începutul anului calendaristic și de numerotarea anilor.

Ziua care marchează începutul anului diferită în diverse epoci și la diferite țări. Bazându-se pe tradiții derivând din ciclul anotimpurilor, cel mai adesea începutul anului era considerat ca fiind la începutul primăverii (în preajma echinoxului). Anglia a mutat începutul anului de la 25 martie la 1 ianuarie abia odată cu adoptarea calendarului gregorian, în 1752. În Franța Primei Republici, anul începea în ziua echinoxului de toamnă.

Numerotarea anilor este o convenție extrem de importantă. În vechime, fiecare țară număra anii de la instaurarea suveranului sau de la un eveniment considerat fundamental. Roma antică număra anii de la fondarea Romei, iar Biserica Catolică a preluat o bună bucată de vreme acest reper. Mai recent, calendarul republican francez a numărat anii de la instaurarea Primei Republici Franceze (1792) dar, fiind adoptat la un an după eveniment, nu există anul I.

În secolul VI, un călugăr pe nume Dionysius Exiguus a scris o istorie a creștinismului (*Cyclus Paschalis*) în care a folosit ca reper temporal anul nașterii lui Iisus Hristos, pe care l-a considerat anul 1. Astfel, a ajuns la concluzia că anul în care și-a scris lucrarea era cel de-al 525-lea al "erei creștine". Sistemul a fost preluat și în alte lucrări, dar semnalul adoptării sale pe scară largă l-a dat împăratul Charlemagne (742-814), care l-a oficializat în imperiul pe care-l conducea.

Pentru ca anul nașterii Mântuitorului să fie singurul reper, sistemul număra anii "erei pre-creștine" pornind tot de la acest an, dar în sens invers. Astfel, anul care a precedat anul 1 a fost considerat anul 1 "înainte de Hristos", precedat de anul 2 și așa mai departe. De remarcat că nu există "anul zero".

Modul de desemnare a "erei" depinde de limbă și epocă. Anii erei creștine au fost desemnați în latină ca "anno Domini", abrevierea A.D. păstrându-se până astăzi în diferite limbi (printre care și engleza). Abrevierea engleză B.C. (*before Christ*) este larg utilizată. În românește se folosesc abrevierile "d. H." sau "d. C" (în funcție de grafia Hristos sau Cristos) pentru era creștină și "î. H." sau "î. C." pentru era pre-creștină. Impunerea acestui sistem referențial de către imperiile europene în țări care nu erau creștine a impus "laicizarea" modului de desemnare a erei. Astfel, în engleză se mai folosesc abrevierile C.E. (*common era*) și B.C.E. (*before the common era*). În România, în perioada regimului comunist, s-au folosit abrevierile "e.n." (era noastră) și "î.e.n" (înaintea erei noastre), care sunt încă larg utilizate.

Pentru desemnarea secolelor se utilizează convenția următoare: secolul I începe în anul 1 d.H. și durează până în anul 100 d.H. (inclusiv); secolul II începe în anul 101 și durează până în 200. Și așa mai departe. La fel și pentru milenii. Așadar, începutul secolului 21 și a mileniului III nu este 1 ianuarie 2000, ci 1 ianuarie 2001. Este însă evident că marea sărbătoare va avea loc la "schimbarea prefixului". Să sperăm deci că *millennium bug* nu ne va umbri sărbătoarea...

O precizare legată de limita inferioară a anilor (`d_min`): ea nu limitează reprezentarea datelor (deci pot fi reprezentate orice date ale erei noastre) ci doar aplicabilitatea unor funcții. Aceasta deoarece în anul 1919 în România s-a adoptat calendarul gregorian, motiv pentru care 13 zile lipsesc practic din calendar, ceea ce complică foarte mult aritmetica datelor.

Alegerea setului de funcții l-am făcut pe baza operațiilor cele mai frecvente care apar în programe. Funcțiile de validare și conversie sunt în general folosite atunci când se preiau date introduse de utilizator sau create de alte aplicații, motiv pentru care am admis un format de intrare foarte liber (procesarea stringurilor se poate dovedi destul de complicată la implementare).

Implementarea funcției `d_Today` va utiliza funcția `GetDate`, motiv pentru care este apelat unit-ul DOS.

Funcțiile aritmetice sunt simple ca interfață, dar implementarea lor este migăloasă. Funcția `d_DifY` s-ar putea să nu vă pară necesară, dar în realitate este foarte adesea utilizată pentru calculul vârstelor. Implementarea ei trebuie să urmeze această logică (de exemplu, dacă `d1` este 12 martie 1998 și `d0` este 13 martie 1960, rezultatul va fi 37). Această funcție ar trebui să funcționeze și asupra datelor dinainte de `d_min`.

Pentru funcțiile de formatare, apelate de regulă la afișarea sau tipărirea datelor, se putea opta pentru un set de funcții (câte una pentru fiecare format), dar am considerat că pentru utilizator este mai simplu să folosească un șablon, cu toate că implementare este mai dificilă.

În fine, tratarea erorilor este o altă problemă spinoasă. Varianta utilizării unui parametru de tip referință pentru fiecare procedură sau funcție ar fi fost o variantă acceptabilă, dar ar fi complicat interfața. Utilizarea unei variabile unice, `d_err`, este suficient de simplă, în condițiile în care fiecare rutină implementată se va îngriji de plasarea valorii corecte.

Codurile de eroare vor trebui documentate pentru fiecare rutină în parte. O convenție folosită adesea este ca pentru fiecare tip de eroare să se atribuie un cod numeric având ca valoare o putere a lui 2, iar la lansarea în execuție a rutinei să se plaseze valoarea zero în `d_err`. Pentru fiecare eroare depistată, codul este *adunat* în `d_err`, astfel încât utilizatorul să poată să descifreze toate erorile depistate. De pildă, în cazul funcției `d_Num`, codurile de eroare ar putea fi:

1: Y < d_min;
2: Y > d_max;
4: M = 0;
8: M > 12;
16: D = 0;
32: D > 31
64: Data inexistentă (ex.: 30 Feb. 1998)

Pentru valorile de intrare Y = 1910, M = 25 și D = 0 se va obține codul de eroare 25 (1 + 8 +16), din care utilizatorul va putea deduce toate cele trei erori.

Desigur, o astfel de bibliotecă de rutine este încă departe de puterea unui tip predefinit. Avantajele față de o abordare improvizată sunt însă evidente: o echipă de programatori poate să implementeze în mod elegant această interfață, o poate testa temeinic și apoi o poate folosi în diverse proiecte. Rezultatele însă depind de o disciplină riguroasă: nimeni nu trebuie să manevreze datele în programul client decât prin funcțiile oferite de unit. Eventualele erori pot fi localizate cu ușurință, iar o re-implementare mai eficientă a interfeței va spori eficiența tuturor programelor care o folosesc, fără a necesita vreo intervenție în codul acestora.

Ce îmbunătățiri și extensii s-ar putea aduce acestui unit? O primă idee ar fi parametrizarea sa pe baza unui fișier de configurare. Aici ar putea, de exemplu, să găsească elemente de "internaționalizare" (numele lunilor și zilelor săptămânii în diverse limbi străine. O altă extensie ar putea fi eliminarea limitei inferioare `d_min` și chiar manevrarea datelor calendaristice dinainte de anul 1. Pentru aceasta este bine de știut că schimbarea calendarului iulian cu cel gregorian în România s-a petrecut în 1919, când după data de 31 martie a urmat data de 14 aprilie.

În fine, o extensie utilă ar putea fi introducerea noțiunii de "zi lucrătoare", deoarece există termene exprimate în acest fel (de exemplu: "în termen de cel mult 10 zile lucrătoare"). În acest caz, fișierul de configurare ar putea conține o listă de zile nelucratoare iar un algoritm special ar trebui să calculeze care sunt zilele de Paște.

"Scheletul" variantei propuse în articol este prezentat în pagina alăturată (fără cod de implementare).

200.000 lei premiu!

Poate că în alte articole voi prezenta implementări ale unor tipuri de date prin clase de obiecte și prin componente. Oricum, intenția acestui articol a fost să evidențieze o problemă uriașă ("scama" mileniului) care putea fi evitată - sau măcar diminuată - printr-o tratare riguroasă din perspectiva unui anumit tip de dată: data calendaristică.

Și pentru că implementarea semanticii acestui tip nu este banală, GInfo oferă un premiu de 200.000 lei pentru cea mai corectă, eficientă, completă și elegantă implementare a unui unit Pascal cuprinzând semantica datei calendaristice. Se poate implementa interfața propusă în acest articol sau oricare altă interfață, cu condiția să fie comodă și funcțională.

Atenție: nu este o problemă didactică, ci una care aparține realității. Criteriile noastre de aprecierul nu vor fi așadar legate nu doar de performanță, ci și lizibilitate, de calitatea documentației etc.

Codul sursă și documentația (doar în format digital!) trebuie să fie expediate până cel târziu în data de 1 iulie 1998. Programul câștigător va fi publicat pe situl Web al GInfo și va fi distribuit ca freeware. Succes!

Câteva adrese:

`http://cssa.stanford.edu/~marcos/ec-cal.html`
(Calculation of the Ecclesiastical Calendar)
`http://www.pip.dknet.dk/~pip10160/`
(The Calendar FAQ)
`http://www.year2000.com/`
(TheYear 2000 Information Center)

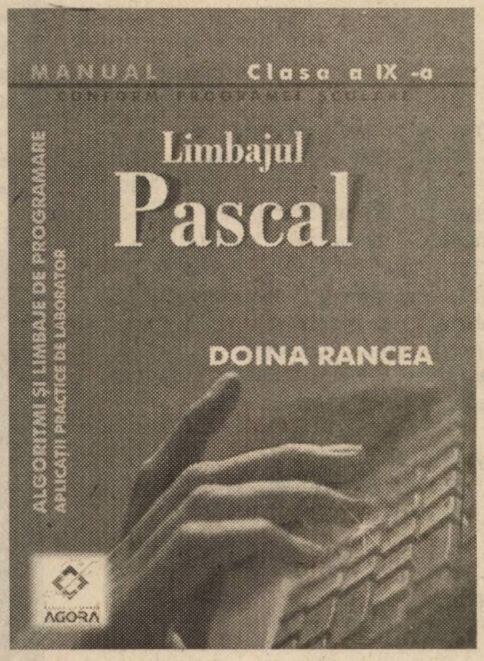
```
UNIT CALENDAR;

INTERFACE

  Uses DOS;

Type Data = String[8];
 { Reprezentarea datei calendaristice prin 8 caractere.}
 { Semnificatia: YYYYMMDD; YYYY= an, MM= luna, DD= zi   }

Var d_err: Byte;
 {variabila va cuprinde codul de eroare a ultimei   }
 {proceduri executate. Valoarea 0 inseamna "corect" }

Const
       d_Min: Word = 1920;
       d_Max: Word = 9999;
       { Anii intre care functioneaza aritmetica    }
       { datelor si functia DayOfWeek.              }

 {--- Functii de conversie si validare ---}
 Function d_YMD(d: String): Boolean;
 { Valideaza o data calendaristica                  }
 Function d_DMY(d: String): Data;
 Function d_MDY(d: String): Data;
 { Produc la iesire o data valida, convertind o     }
 { data exprimata intr-un alt format (conform       }
 { numelui procedurii). Caracterele care nu sunt    }
 { cifre sunt considerate separatori.               }

 Function d_Num(Y, M, D: Word): Data;
 { Produce la iesire o data valida, pe              }
 { baza parametrilor primiti                        }

 Function d_Today: Data;
   { Produce la iesire data curenta, conform datei  }
   { din sistem}

 {--- Aritmetica datelor ---}
 Function d_DifD(d1, d0: Data): integer;
 Function d_DifY(d1, d0: Data): integer;
   { Produc la iesire diferenta intre datele d1 si  }
   { d0, exprimate ca numar intreg de zile si,      }
   { respectiv, ani. Daca data d1 este mai mare,    }
   { rezultatul este pozitiv, altfel este negativ.  }

 Function d_AddD(d: Data; x: integer): Data;
   { Produce la iesire data care va fi la x zile    }
   { dupa data d (daca x > 0) sau data care a       }
   { fost cu x zile inaintea datei d (daca x < 0).  }
   { Daca x este 0,  produce data d.}
 Function d_Calc(d1: Data; Y, M, D: Integer): Data;
   {Produce la iesire o data obtinuta adunind la    }
   { d1 numarul Y ani, M luni si D zile. Y, M       }
   { si D pot fi si negative.                       }
   { Ex: '19960915' + (3, 3, 20) = '19991215' +     }
   { + 20 zile = '20000104'                         }

 {--- Functii de formatare ---}
 Function d_Ext(d: Data; pattern: String): String;
   {Produce o reprezentare a datei d pe baza
    unui sablon. In sablon se fac inlocuirile:
   - "d" - ziua - exprimata cu minimum de cifre
   - "D" - ziua - exprimata cu 2 cifre
   - "m" - luna - exprimata scurt
   - "M" - luna - exprimata complet
   - "n" - luna - exprimata numeric (minimum de cifre)
   - "N" - luna - exprimata numeric cu 2 cifre
   - "y" - anul - exprimat cu 2 cifre
   - "Y" - anul - exprimat cu 4 cifre;
   Toate celelalte caractere din sablon sunt pastrate
   ca atare in reprezentare.
   Exemplu: 981001 -> "m d, Y" = "oct 1, 1998"
                   -> "D-N-y"  = "01-10-98" }

 Function d_DayOfWeek(d: Data): String;
  {Produce ziua din saptamana corespunzatoare datei d}

IMPLEMENTATION

  Const
       Short: array [1..12] of String[3] =
         ('ian', 'feb', 'mar', 'apr', 'mai', 'iun',
          'iul', 'aug', 'sep', 'oct', 'nov', 'dec');
       Long:  array [1..12] of String[10] =
          ('ianuarie', 'februarie', 'martie',
          'aprilie', 'mai', 'iunie', 'iulie',
          'august', 'septembrie', 'octombrie',
          'noiembrie', 'decembrie');
       Week: array [0..6] of String[8] =
            ('duminica', 'luni', 'marti', 'miercuri',
             'joi', 'vineri', 'sambata');

      Days: array [1..12] of Byte =
            (31, 28, 31, 30, 31, 30, 31, 31, 30,
             31, 30, 31);

  Type
      Normal = record
         Y, M, D: Word;
         end;
      Julian = record
         Y, D: Word
         end;

  Var
     n: Normal;
     j: Julian;

  { Implementarea procedurilor locale }

  Function StrToNum(Str: String): Word;
  Function NumToStr(n: Word): String;
  { Transforma stringuri in numere si invers }

  Procedure To_Normal(d: Data; Var n: Normal);
  Procedure To_Julian(d: Data; Var j: Julian);
  {Convertesc datele calend. in formate interne }

  Function Leap(Y: Word): Boolean;
  { Stabilesta daca anul este sau nu bisect}

  { === Implementarea interfetei =====}
  Function d_YMD(d: String): Boolean;
  ... etc. etc. etc....
```

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie*

... un MANUAL.



Limbajul Pascal este un manual care acoperă programa școlară (1997) a două discipline: „Algoritmi și limbaje de programare" și „Aplicații practice de laborator" care se studiază în clasele a IX-a cu profil de informatică.

Manualul prezintă noțiunile din programă la un nivel accesibil, fiecare noțiune fiind ilustrată în mai multe programe Borland Pascal 7.0. Aceste programe sunt însoțite și de prezentarea datelor de intrare și a rezultatelor în forma sub care apar pe ecran. Problemele au fost selectate în așa fel încât să ilustreze multiplele domenii de utilizare a calculatoarelor. Nu lipsesc problemele de simulare, jocuri, probleme distractive, dar au fost alese și aplicații din programa de matematică de clasa a IX-a cu scopul exemplificării și fixării unor noțiuni importante din matematică. În manual se insistă asupra felului în care se descoperă rezolvarea unei probleme, nivelul de dificultate a acestora crescând gradat, venind astfel în sprijinul elevilor și profesorilor.
Numeroasele probleme din manual permit utilizarea acestuia nu numai pentru studiu individual ci și ca bogat material didactic pentru lecțiile de programare și laborator.
Manualul se adresează elevilor, studenților, profesorilor și oricărei persoane interesate de programare.
Manualul a fost conceput, plecând de la ediția din 1993 (Limbajul Turbo Pascal, clasa a X-a) care a fost regândit substanțial și îmbogățit, astfel încât să fie accesibil unui elev de clasa a IX-a. Ediția anterioară s-a bucurat de un succes deosebit, fiind apreciată de mii de cititori.

Autoarea, Doina Rancea, este absolventă cu diplomă de merit a Facultății de Matematică, Secția Informatică a Universității Babeș-Bolyai din Cluj Napoca (1975). A fost repartizată la Liceul de Informatică din Cluj Napoca unde predă și în prezent limbajele de programare Pascal și C++. Este profesoară gradul I și în această calitate îndrumă activitatea de practică pedagogică a studenților. Experiența bogată a autoarei a fost valorificată cu ocazia participării, în cadrul Comisiei Naționale de Specialitate Informatică a Ministerului Educației Naționale, la actualizări ale programei școlare pentru disciplina informatică și în cadrul comisiilor județene și naționale ale olimpiadelor, propunând probleme originale.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Poșta redacției

De data aceasta, am selectat mesaje din arhiva listei dialog. Acolo forma de dialog e mai directă, se pot da citate, se dau replici etc. Din acest motiv a trebuit să recurg din nou la numerotarea paragrafelor la care se face referire și, uneori, să reamintesc [între paranteze drepte] ideea. Sper să vă placă. Nu uitați, vă invit în continuare la <dialog>.

Notă: Deoarece mesajele au fost preluate de pe e-mail, diacriticele au fost adăugate de mine. E foarte probabil că am scăpat câteva. Vina îmi aparține.

From: aastefan@mail.dntis.ro
To: msarbu@agora.ro
Subject: critici si sugestii

Salut Mircea,

(1) [...] Prima impresie nu a fost bună. Cu câteva excepții, totul a fost o lectură (plăcută, trebuie să recunosc), dar o simplă lectură.

(2) Eu folosesc calculatoarele din clasa I (când aveam un XT) și multe lucruri îmi erau cunoscute. Mă refer aici în special la "Scrisori în eter" și "Povestea lui Linus Torvalds".

(3) "Cei care au creat Java" ar fi fost interesant, dar cu câteva luni înainte citisem un articol în Chip intitulat "Cum s-a născut Java", articol care semăna cu ceea ce s-a publicat în revistă.

(4) "Adevărul despre LOGO" a fost bun și s-a referit la un subiect mai puțin cunoscut, dar totuși interesant. La fel gândesc și despre "Fotbal, software și roboți".

(5) Ați ajuns la episodul 2 dedicat limbajului Java. Mi-au plăcut cele două articole, dar cred că ar trebui să opriți acest serial din cel puțin două motive:

1. Cei cărora nu le place sau nu programează în Java se vor plictisi.

2. Cei care s-au apucat să învețe Java nu cred că au răbdare să aștepte o lună ca să primească o nouă lecție, ci mult mai probabil își vor cumpăra o carte despre Java.

(6) Aș vrea să citesc în GInfo, articole noi, originale. În acest sens, articolele "Durata de execuție a algoritmilor" și "Algoritmul Boyer-Moore" au fost grozave și de aceea vă rog să prezentați și alți algoritmi.

(7) Mi-au plăcut de asemenea problemele rezolvate.

(8) Mă opresc aici. Sper să te fi ajutat cu sugestii utile.

[Ștefan Atanasiu]

(1) Îmi amintesc când am citit prima carte americană de informatică (Patrick Winston: "Inteligența Artificială" - apărută, prin nu știu ce minune, la Editura Tehnică în anii '80). Am fost șocat de stilul colocvial al autorului (un "guru" al domeniului). Eram obișnuit cu stilul doct (adesea "scorțos") al majorității lucrărilor de specialitate pe care le citisem până atunci... Cartea lui Winston era o lectură în același timp plăcută și utilă. Cu atât mai mult o revistă, cred, trebuie să fie o lectură plăcută. Deci, până aici, e bine.

Că GInfo trebuie să fie mai mult decât o simplă lectură, e la fel de adevărat. Dar, încă o dată vreau să vă atrag atenția că GInfo nu este o anexă a programei școlare, nu este un "manual foileton", nu este "caiet de informatică pentru clasele IX - XII" sau ceva de acest gen... Este o revista, ceea ce implică de la sine un caracter fragmentar, caleidoscopic, divers și... plăcut.

(3) Aceeași poveste, oricum ar fi spusa, rămâne în esența la fel. Sursele de informare sunt aceleași (amintirile celor de la First Person). Nu știam de articolul din Chip.

(4) E un punct de vedere asupra căruia vom medita. Într-un fel, motivațiile stau în picioare... Pe de altă parte, cred că programarea în Java nu poate să lipsească din GInfo. Mă voi consulta cu dl. Horia Georgescu pentru a găsi o varianta care să funcționeze (eventual un serial care să nu fie un "curs de inițiere", ci care doar punctează aspecte mai interesante... sau algoritmi... sau implementări...).

(6+7) Mă bucur că ți-au plăcut și cu siguranță vei mai găsi așa ceva. Ar fi însă de discutat o problemă: din câte constat, în școală se pune un accent foarte mare pe algoritmică. Ceea ce, până la un punct, este foarte normal. Dar algoritmica nu înseamnă decât o mică parte din tehnica programării... Există nenumărate aspecte legate de analiza problemei, de arhitectura aplicației, de implementare, de testare, de întreținere, de lucru in echipă, de tehnici formale sau informale de specificare... Acestea nu prea se învață la școală (și uneori nici la facultate) dar sunt extrem de utile. Și interesante (desigur, nu vreau să aduc în revistă plictisul inevitabil al programării ca muncă de rutina)

(8) Cu siguranță.

From: mairon@radiotel.ro
To: <msarbu@agora.ro>
Subject: Re: Replica pt. Dl. Stefan Atanasiu

Stimate D-le Sârbu,

Am primit prin "dialog" scrisoarea D-lui Atanasiu și aș vrea să vă transmit și eu câteva opinii.

Așa cum i-am transmis și D-lui Laslo Szabo, părerea mea despre GInfe este excelentă!.

Este cu cel puțin o clasă peste vechea serie. În plus, exista facilitatea cu "dialog" și pagina web.

Dar haideți să comentez și eu scrisoarea D-lui Atanasiu.

[...] (2) "Scrisori în eter" a fost un articol excelent și foarte util. Dovada este scrisoarea D-nei Prof. Magdalena Stoica. "Povestea lui Linus Torvalds" a fost interesantă iar datele de evoluție Linux foarte utile.

(3) Articolul din CHIP a fost mult mai restrâns și mai slab!

(4) Serialul despre Java este excelent! Nu faceți greșeala să îl opriți!

[...] Iată argumentele mele în favoarea continuării serialului:

Java este limbajul viitorului și deja în afară se "pedalează" puternic pe el. Dl.Atansiu va pierde trenul. :-)

Java are vârsta de doar 2 ani deci nu

există prea multa documentație serioasa despre el.

Puțini își permit să cumpere singura carte serioasă despre Java existentă la noi - a lui Peter Norton (190.000 lei) și deja este cam veche:1996.

Dl. Georgescu prezintă excelent limbajul într-o abordare comparativă. Și apoi domnia sa nu își propune să ne învețe Java, ci numai să prezinte câteva aspecte esențiale.

Domule Sârbu, va mai aduceți aminte câte scrisori ați primit în Byte de genul: "Opriți articolele despre Internet! Nu sunt valabile la noi!" Acum câte mai primiți? Cred că foarte puține. Cam așa sună și rugămintea D-lui Atanasiu.

Dacă totuși sunteți de părerea d-lui Atanasiu, va rog totuși să supuneți la vot oprirea serialului.

(6+7) Am aceeași părere. Dl. Atanasiu îmi devine simpatic. :-)

Cu stima,

Ing. Adrian Tufa

```
From: aastefan@mail.dntis.ro
To: dialog@ginfo.ro
Subject: Re: Replica pentru Dl.Atanasiu
```

Îmi permit să comentez ultima scrisoare a dl. Tufă.

[...] (2) "Scrisori în eter" a fost un articol bun, dar cu lucruri pe care eu (și probabil și alți cititori) le cunoșteam. Cât despre "Povestea lui Linus Torvalds": nu știu daca ați lucrat în Linux. În majoritatea distribuțiilor exista un fișier html în care Linus prezintă istoria sistemului de operare pe care l-a creat. Datele de evoluție se găsesc și aici, și sunt chiar mai detaliate decât în revistă. Același fișier se găsește și pe Internet.

(3) [Articolul din CHIP] Mai restrâns da, mai slab nu.

(4) [Serialul despre Java este excelent!]. De acord.

[Nu faceți greșeala să îl opriți!] Nu chiar acum. Nu uitați că acest serial își propune să trezească interesul cititorului pentru limbajul respectiv, să-l ajute să creeze primele sale aplicații, să prezinte aspectele fundamentale al limbajului în cauză. Atât și nimic mai mult. În nici un caz nu ar trebui transformat în curs de programare.

[nu există prea multa documentație] Există documentație serioasă. Chiar dvs. afirmați asta mai jos !!!

Și totuși, dacă nu ar exista documentație, trebuie ca Ginfo să scrie o astfel de documentație??

[Puțini își permit să cumpere...] Adevărat. Dar există alte soluții. Există biblioteci, exista anticariate, există Internet, exista prieteni de la care se pot împrumuta cărți.

[Dl. Georgescu prezintă excelent limbajul...] De acord.

[... nu își propune să ne învețe Java...] Aici vroiam să ajung și eu. Nu poți învăța Java dintr-o revistă. Revista poate să-ți trezească interesul pentru acest limbaj. Și aici cred că GInfo și-a atins scopul.

[..."Opriți articolele despre Internet!] Din nou m-ați înțeles greșit. Sau poate n-am înțeles eu ce înseamnă acel "cam".

[... supuneți la vot...] Mi se pare corect.

(6+7) [Dl. Atanasiu îmi devine simpatic. :-)] Sentimentul e reciproc :)

[Stefan Atanasiu]

Trebuie să recunosc că m-a amuzat schimbul de replici. Nu a fost chiar un "flame war", dar au sărit câteva scântei... Vreau să mai fac câteva precizări pe marginea celor discutate.

(2) Există **enorm** de multă informație pe ambele subiecte (e-mail și Linus). Pentru cei care au ajuns să cunoască lucrurile, este evident că articolele sunt interesante cel mult din punct de vedere... stilistic. Însă pentru ceilalți, această uriașă cantitate de informație este un handicap. Cu ce să începi? Care sunt lucrurile fundamentale și care sunt amănuntele? Tocmai aici intervine rolul nostru: această operație de sistematizare și condensare nu este deloc ușoară. Însă dacă este bine făcută, este un punct de referință pentru cititor, un reper de la care poate să pornească în orice direcție.

(4) Serialul despre Java va continua. Aprecierile sunt unanime (chestionarele pe care le-am primit confirmă acest lucru). Sunt convins că Profesorul Georgescu, având experiența multor ani de catedră, știe foarte bine cum și unde să-l conducă pe cititor.

Pe de altă parte, există multă documentație despre Java, chiar și în românește. Și va mai apărea (din surse confidențiale știu că foarte curând va apărea o nouă carte - abordare excelentă, sub girul unor nume de marcă a informaticii românești). Găsiți chiar și un manual *on-line* în limba română (http://www.integrasoft.ro/cursuri.html - versiunea electronică a cărții lui Eugen Rotariu publicată la Computer Press Agora în 1996). Cei ce nu au răbdare să aștepte câte o lună pentru fiecare lecție au toate posibilitățile să studieze singuri.

Cât despre vot... Nu e posibil și nici nu e cazul.

```
From: Catalin Drula <catalin@lbi.ro>
To: dialog@ginfo.ro
```

(9) Eu nu prea înțeleg... cui se adresează "Gazeta de Informatică"? Adică, eu credeam ca era o revistă adresată în special elevilor și studenților, cu probleme de informatică, algoritmi, rezolvări etc. Și revista se adresa celor "peste medie": olimpici s.a... Deci nu chiar "geniilor", dar nici "pentru to(n)ți"! Cel puțin așa era formatul vechi și la fel sunt și "Gazeta de matematică", "Gazeta de fizică".

(10) Dvs. ați "deturnat" revista oarecum... (prin subiectele pe care le abordați). Poate greșesc, dar dacă am dreptate și revista nu se mai încadrează în prima descriere pe care am dat-o, atunci a rămas "Gazeta de informatica" doar cu numele. Eu înțeleg că trebuie să o vindeți și de-asta vă adresați unui public mai larg, dar în cazul asta trebuia să lăsați "Gazeta" cum era înainte (bănuiesc că subvenționată, nu?) și să faceți o revistă cu alt nume și cu conținutul actual.

[Catalin Drula, elev, cls IX, Liceul de Informatică București]

(9) Este corect ceea ce spui. Problema nivelului de accesibilitate a revistei este un permanent subiect de dispută. Eu cred că se va regla de la sine, cu condiția să discutăm.

(10) Oarecum. În sensul că am lărgit puțin tematica. Am constatat ca sunt foarte mulți elevi și studenți care nu sunt interesați **doar** de algoritmica. Dacă vrei să te convingi, urmărește puțin suplimentul PC CONCRETe al revistei PC Report (eventual pe Web: http:\\www.pcconcrete.ro, sau abonament gratuit prin e-mail). Vei recunoaște foarte multe adrese ale liceelor.

Dar, observă ca tematica "vechii" Gazete de Informatică este continuată. Găsești probleme rezolvate, probleme

propuse, articole de algoritmică etc., la același nivel calitativ ca și înainte. De fapt e aceeași redacție (puțin lărgită), aceeași tematică (puțin lărgită), deci aceeași revistă (puțin lărgită). Simte-te deci în largul tău, e loc pentru toată lumea :-).

O observație: Gazeta de informatică nu a fost niciodată subvenționată. A fost făcută cu pasiune și cu un efort demn de toată admirația, nu întotdeauna răsplătit altfel decât de satisfacția muncii bine făcute și a aprecierii cititorilor.

Fii mai explicit: ce anume te nemulțumește? Scopul acestui dialog este exact acesta: să ne lămurim unii pe alții. Noi, redacția, să înțelegem ce anume ați cere voi de la revistă, iar voi, cititorii, să înțelegeți cât de diverse pot fi cerințele pe care încercăm să la satisfacem.

From: Catalin Drula <catalin@lbi.ro>
To: dialog@ginfo.ro
Subject: Re: Alta provocare

Salut din nou.

Îmi cer scuze pentru tonul puțin iritat al mesajului anterior. Ce am vrut eu era să aflu "ce e" cu revista asta. Să înțeleg exact ce vrea și cui se adresează.

(11) Am văzut că ați lansat provocarea "revistă elitistă" contra "revistă populară". Bănuiesc că m-aș putea considera în categoria elitiști (și eu am fost la Olimpiada Națională, locul 4, premiul I). Totuși acum mă simt prost cerând o revistă pentru elite când (probabil) că sunt atâția care apreciază revista în forma populară, care au de învățat din ea chiar în forma asta, care sunt interesați de revistă și care și-ar pierde interesul dacă ar deveni "elitistă".

(12) Sunt însă puțin dezamăgit pentru că am crezut că revista asta va fi... cum sa spun, un mediu pentru schimbat idei, probleme etc. al celor din elită. Pentru că astfel de medii există în afară; chiar un fel de societăți naționale (ACM, IEEE) care scot diferite jurnale (Journal of Graph Theory, Journal of Algorithms s.a.m.d.) unde oamenii își prezintă cele mai recente descoperiri, algoritmi... Un loc comun pentru vârfurile CS din acea țară. Deci asta credeam eu că o să fie Gazeta de Informatică (chiar îmi propusesem să public un articol-două). Să nu se înțeleagă că vă acuz pentru că nu asta este revista. Vroiam doar să îmi explic poziția. Greșeala îmi aparține.

(13)Apropo de algoritmică... Cum am mai spus, fără teorie, baze matematice, algoritmi, nu ar fi existat nici calculatoare, nici compilatoare, nici sisteme de operare. Toate astea au avut întâi o dezvoltare teoretică (chiar și în proiectarea procesoarelor se folosesc algoritmi din geometrie analitică de ultimă oră). Și domeniul nu e mort. Adică cel puțin asta e părerea mea.

(14) Spuneți că majoritatea programatorilor nu știu algoritmică. Nici cel care face o aplicație în FoxPro, nici cel care scrie un applet Java, nici cei de la Microsoft.

(15) Aș putea să vă răspund că poate dacă știau cei de la Microsoft algoritmică, ar fi scos și ei software de calitate și fără atâtea buguri. Să nu mai vorbim de nevoia de resurse din ce în ce mai mari. Mai mult RAM, mai mulți Mhz, mai mulți GB de disk. S-a pierdut spiritul acela de a face totul cât mai eficient, cu cicli de procesor cât mai puțini, cu memorie cât mai puțina. Mă întreb câți programatori de la Microsoft știu ce înseamnă complexitate. Pe de alta parte, în Linux s-a păstrat spiritul acesta, de a face totul cât mai eficient (chestie care se vede dacă urmăriți lista linux-kernel o săptămâna). Și diferența față de SO Microsoft zic eu că se vede, nu?

(16) Oricum ar fi multe de zis pe subiectul acesta și probabil că nu sunt eu în măsură să-mi dau cu părerea, mai ales că nici nu m-am născut odată cu calculatorul, ca să pot să urmăresc evoluția IT. Sunt doar părerile unui elev de clasa a XI-a...

(Cu respect, Cătălin Drula)

(11) Încercăm să "îmbinăm" puțin extremele. Orice ziar sau revistă iei în mână, constați că nu te interesează **totul**. Este inevitabil. Așa va fi și cu GInfo. Poate că cele două articole pe care le consideri "grozave" împreună cu încă vreo două care îți releva niște tematici noi, să merite osteneala și banii. Poate ca nu...

(12) Aș vrea să lămurim puțin problema aceasta. Am primit o vreme o revistă care se chemă "American Programmer". O revistă alb-negru (inclusiv coperta), pe hârtie reciclată, vreo 50 de pagini format mic, 10 numere pe an. Foarte interesantă (numere tematice: ISO 9000 în software, hackers etc). Dar... abonamentul anual costa 120 US$!!! Adică 12 dolari numărul (cca. 100.000 lei), în condițiile în care PC Magazine (de pildă), la vreo 300-400 de pagini, costă vreo 3 dolari.

Un aspect economic: costurile tipografice se distribuie per exemplar tipărit. Filmele și plăcile tipografice costă la fel, oricâte exemplare tipărești. E clar, nu? Ai fost la ONI la Oradea: erați vreo 150 de olimpici. Să admitem că asta e elita. O revistă în 150 de exemplare nu are sens.

Pe de altă parte: revistele pe care le-ai menționat sunt "academice". Aceasta înseamnă un cu totul alt "flux de producție", alte canale de distribuție, alt mod de finanțare. Cu toată sinceritatea îți spun că nu vrem să facem așa ceva.

Dar: am putea face (with a little help from my friends :-) un colț "elitist" în pagina Web. Aici am putea pune articole pe care noi, redacția, le considerăm de nivel prea "înalt" pentru publicare în revista propriu-zisă. Sunt sigur că am putea obține colaborarea unor profesori din învățământul liceal și universitar, care să joace rolul de recenzenți. Aceste pagini ar putea fi distribuite și prin e-mail, pe baza unui abonament (poate gratuit, poate în conexiune cu abonamentul la GInfo...). Crezi ca ar fi interesant? Poate din toamnă...

Cât despre articole... Oricând ești binevenit.

(13) Doamne ferește! Cum să fie mort?!?

(14) Pardon, nu am afirmat așa ceva!!! Am zis doar că **există** programatori care nu știu algoritmică. Chiar am adăugat că "nu programatori buni". De aici și până la afirmația pe care mi-o atribui e o cale așa de lungă...

(15) Hmm... Am rețineri. Programatorii de la Microsoft nu știu algoritmică și nu știu ce înseamnă complexitate? Windows NT nu este un produs de calitate ? Nu sunt un fan al produselor Microsoft de larg consum, dar... Pe de altă parte, accentele de "fundamentalism" ale linuxiștilor nu prea mă încântă.

(16) E ok, lăsăm subiectul acesta pentru altă dată...

Toate cele bune și pe curând,

Mircea

Vă place GInfo?

Nume: _____ Prenume: _____ Vârsta: _____

Adresa: _____

Localitate: _____ Cod poștal: _____ Județ: _____ Telefon: _____

Ocupația: elev informatică ☐ elev ☐ student ☐ cadru didactic ☐ altă ocupație ☐ (precizați: _____)

Dotări

Folosiți un calculator acasă? Da ☐ Nu ☐

Câte calculatoare sunt instalate în școală: (aprox.) _____

Aveți acces la e-mail: Da ☐ Nu ☐

Aveți acces la WWW: Da ☐ Nu ☐

Cum apreciați dotarea cu tehnică de calcul a școlii: _____

Aprecieri generale

Acordați note de la 1 la 10 pe următoarele criterii:

Structura tematică: ___

Calitatea articolelor: ___

Prezentare grafică: ___

Calitate tipografică: ___

Considerați că prețul este:

prea mare ☐ accesibil ☐ prea mic ☐

De cât timp cunoașteți Gazeta de Informatică (ani)? _____

Cum ați procurat acest număr:

chioșc ☐ abonament ☐ cadou ☐ altfel ☐

În acest număr

Apreciați cu note de la 1 la 10 articolele acestui număr:

Un algoritm de factorizare ___

Turnurile din Hanoi în Java ___

IRAM sau memoria inteligentă ___

Millennium Bug ___

Numere Fibonacci ___

Principiul inducției în informatică ___

Povestea OS-ului ___

Articolul care v-a plăcut cel mai mult: _____

De ce? _____

Articolul care v-a plăcut cel mai puțin: _____

De ce? _____

Considerați că ponderea problemelor rezolvate este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea problemelor propuse este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea subiectelor teoretice este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că nivelul de accesibilitate al articolelor este

prea cobotât ☐ potrivit ☐ prea ridicat ☐

În viitor

În următoarele numere așteptați:

mai multe articole de programare? Da ☐ Nu ☐

mai multe articole teoretice? Da ☐ Nu ☐

mai multe articole despre Internet? Da ☐ Nu ☐

mai multe prezentări de programe? Da ☐ Nu ☐

prezentări de jocuri? Da ☐ Nu ☐

articole despre hardware? Da ☐ Nu ☐

V-ar plăcea să citiți:

pagini redactate de elevii unor licee? Da ☐ Nu ☐

lucrări premiate, realizate de elevi? Da ☐ Nu ☐

actualități școlare? Da ☐ Nu ☐

prezentarea unor situri Web? Da ☐ Nu ☐

tips & tricks? Da ☐ Nu ☐

Comentarii:

_____

_____

_____

_____

_____

_____

_____

Dacă nu vă este suficient spațiul, folosiți coli albe suplimentare

EȘTI PREGĂTIT PENTRU MÂINE?

ARE YOU READY FOR TOMORROW?

Premiul 1

COMRACE COMPUTERS îți oferă:

COMRACE ELITE

Intel Pentium 200 MMX

• Placă de bază: VIA MMX PCI P5, 75-233 • 256 Kb PB cache • 16 MB EDO RAM • HDD WD-Caviar 2 GB U/DMA33 • FDD Sony 3,5" • 16x CD-ROM Drive FUNAI EIDE • Microfon, MEGAWAVE Sound-Card 3D YAMAHA F718 SW/WT, Set de boxe SYNTONY 100W 3D (PMPO) • SVGA S3 VIRGE/DX 3D 4 MB EDO RAM MPEG MS-385 • Monitor ADI PROVISTA E30 14" Digital, 30-50Khz, 1024x768, NI, LR, MPR II • Fax-modem BTC 28800 intern V34 MNP2-5 • Tastatură COMRACE 104 Win95 • Mouse COMRACE 3 butoane 400dpi+ Pad • Windows 95 OSR2-PE + IE 4.0 (2CD) • Works 4.5 (CD) + RAV 5.22

ARE YOU READY FOR TOMORROW?

Premiul 2

AGER BUSINESS TECH îți oferă:

3D ComboTV

PIXELVIEW

• Placă PV-CL5465P (4MB RAMBUS, PCI True 3D Accelerator, CL-GD5465 Laguna 3DA, TV-tuner on board, Video Conference ready) • Cameră Video + cablu • Video Phone S/W (Software for Video Conference • Remote Master (Programmable Remote Control) • Microfon unidirecțional •Teletext (PAL)

ARE YOU READY FOR TOMORROW?

Premiul 3

M.G.T. Trade îți oferă:

DVD - ROM Drive

GD-2000 HITACHI

• Sistem DUAL-FOCUS LASER - standard DVD media multi-layer • Timp de acces mediu: 210 ms (DVD media) • Rata de transfer: 2.76 MB/s DVD, 1.29-3.0 MB/s CD-ROM • Interfață IDE (ATAPI) • Compatibilitate CD-ROM, CD-R și CD-RW

ARE YOU READY FOR TOMORROW?

Premiul 4

ASTRA - 1210P

UMAX

•Rez.optică: 600dpi/ Rez.max.:9600dpi • Unică trecere • Viteză de scanare: Gray-4,9ms/lin.; Color-14,7ms/lin. • Arie max.:216x297mm

GRUP TRANSILVAE îți oferă:

SCANNER UMAX

ARE YOU READY FOR TOMORROW?

Premiul 5

NETWORKS îți oferă:

V.O.Tools 2.0

• CD+Manual • Softul care îți oferă mijloacele cele mai simple pentru a crea aplicații care rulează sub Microsoft Windows

ABONEAZĂ-TE ȘI VEI FI!

Pentru abonament folosiți cartea poștală din revistă.
Dintre toți abonații înregistrați până la 10 iunie 1998, se vor trage la sorți câștigătorii celor 5 premii prezentate mai sus.

HEWLETT® PACKARD

Seria de imprimante HP DeskJet 690 cu PhotoREt. Imaginea afacerilor rentabile.

Legenda spune că o imprimantă nu poate însufleți imaginile. Datorită unei strălucite combinații de tehnologii avansate, seria de imprimante HP DeskJet 690 spulberă odată pentru totdeauna această prejudecată. Imaginile imprimate au calitate fotografică datorită tehnologiei HP PhotoREt. Sistemul RealLife imaging asigură, cele mai vii culori şi cea mai bună calitate pentru reproducerile alb-negru. Mai mult decât atât! ColorSmart reglează automat nivelurile de culoare. Cu cartuşul Photo Cartrige, se obțin cele mai naturale culori, iar imaginile tale beneficiază de o calitate fotografică excepțională şi de o mai mare rezoluție. Nici n-apucă bine balaurul să pocnească din coadă şi documentele tale sunt deja tipărite, 5 pagini alb/negru sau 2 pagini color pe minut! Imprimanta HP DeskJet 690C reproduce imagini atât de reale de parcă simți arsura flăcărilor scoase de balaur. Aşa că, uită poveştile! Seria DeskJet 690 nu este una oarecare. Este sufletul documentelor tale!

IMPRIMANTELE HP. HÂRTIA ESTE ÎN SLUJBA TA.

DEALERI – Bucureşti: AMERILEX - 223 43 23 • MBL COMPUTERLAND - 336 75 99 • Q'NET - 211 78 55 • S&T ROMANIA - 330 73 20 • SYSTEM PLUS - 210 85 70 THEMA - 231 29 55 • **Bistrița:** LOGITEC COMPUTER - 063/23 27 96 • **Cluj:** SISTEC-CLUJ - 064/19 02 82 • **Timişoara:** BRIDGEMEN - 056/20 13 60
RESELLERI – Bucureşti: ALSYS DATA - 411 27 27 • BLUE RIDGE INTERNATIONAL - 210 68 28 • CIPRO - 220 04 00 • FLAMINGO COMPUTERS COM - 222 49 34 • FOCUS INTERNATIONAL - 322 30 69 NET CONSULTING - 638 33 45 • PRODUCTON - 312 30 78 • RADIX - 410 25 25 • SOFTWARE ITC - 232 44 71 • TRICORP ELECTRONICS - 679 48 79 • TOTAL SOFT - 230 09 76 **Baia Mare:** CUMULUS LOGIC - 062/43 07 35 • **Bacău:** MIDO IMPEX - 034/17 10 80 • **Braşov:** ICCO COMPUTERS - 068/14 43 54 • INTELPROF INTERNATIONAL - 068/41 13 54 SISTEM - 068/31 86 63 • **Cluj Napoca:** AT COMPUTER - 064/41 32 04 • **Craiova:** AS COMPUTERS - 051/16 32 80 • **Constanța:** FORTE DIGISOFT - 041/65 30 36 • GMB COMPUTERS - 041/61 92 22 **Deva:** ASCOPIER - 054/23 36 06 • **Galați:** ROMLOTUS - 036/47 40 00 • SPARK - 036/46 52 70 • **Hunedoara:** MEMORII TECH DATA SERVICE - 054/23 07 03 • **Iaşi:** RADIX - 032/ 21 24 53 • SERCOM - 032/12 25 89 **Piatra Neamț:** SOFT COMPREST - 033/ 22 21 21 • **Ploieşti:** PLATIN SISTEMS - 044/19 18 18 • **Reşița:** ELCOM - 055/22 07 11 • **Suceava:** PRONOVA SMART - 030/52 14 85 **Timişoara:** ETA-2U: 056/19 95 08 • NETRING COMPUTER - 056/13 28 94 • **Tg. Mureş:** ELECTRO ORIZONT - 065/16 65 72 • **Tulcea:** MICROLINE - 040/51 63 69
CENTRU DE SERVICE AUTORIZAT: 330 73 00

Gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

Vara 1998
Vol. 8/4-5

Lei 8000

din sumarul numărului:

ARITMETICA NUMERELOR MARI

10000

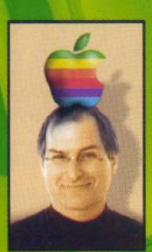
TEHNO
MEMORIA VIRTUALĂ

SUNETUL
o necesitate

MATE
ALGORITMI GEOMETRICI ELEMENTARI

SERIAL
LINUX pentru toți

PORTRET
Povestea MĂRULUI



Probleme propuse și soluții **PLUS** **INTERVIU** și altele

T.P.
Aprobat R.A.P.R.
Nr.103/DC/053/1998
Valabil - 1998 -

INFADRES

Ştiţi cât costă un IBM PC?

doar 1100 USD !!!

Dacă pe lângă cei **800** USD cât costă un pachet **LOTUS SmartSuite** mai plătiţi **300** USD, primiţi şi un calculator original **IBM PC300GL**, în următoarea configuraţie:

- procesor INTEL Pentium MMX 200MHz
- 16MB RAM
- 2,1GB HDD
- placă video PCI 1MB VRAM
- monitor IBM G42 14"
- tastatură IBM 104 taste UK
- mouse IBM
- Windows 95 preinstalat, cu licenţă
- licenţă LOTUS SmartSuite

şi nu este singura noastră ofertă !!!

IBM® Business Partner

Microsoft
WHERE DO YOU WANT TO GO TODAY ?™
Authorized Distribution Partner/
DSP for Romania

FORTE COMPUTERS
Flexibility.Made in Romania

IBM® Business Partner

Lipscani 102, Bucureşti
Tel: 3122360/6137282/3120948 Fax: 3122630

Vol. 8, Nr. 4-5 1998

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu* (București)
Redactor șef:
prof. *Clara Ionescu* (Cluj)
Editor executiv:
Mircea Sârbu (Tg-Mureș)
Au mai colaborat:
Lucian Vințan (Sibiu)
Grigore Albeanu (București)
Gabriel Prișcanu (București)
Tiberiu Socaciu (Sibiu)
Ioan Cozac (Târgu-Mureș)
Dragoș Marian (Târgu-Mureș)
Horia Giurgiu (Cluj-Napoca)
Cristian Mateescu (București)

Tehnoredactare:
Călin Borz
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
C.P. 172-1, 4300 Tg-Mureș
Adresa e-mail: office@ginfo.ro
Tel: (065) 166.516
Fax: (065) 166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *S.C. Infopress srl*,
Odorheiu Secuiesc,
tel: 066.216.483
Editor: Computer Press Agora srl
str. T. Vladimirescu nr. 63/1
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director editorial:
Mircea Sârbu
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: (01) 330.92.81

Sumar

soluții.. PAG. 4

probleme REZOLVATE
PAGINI REALIZATE DE CLARA IONESCU
Probleme propuse spre rezolvare în GInfo nr. 2

probleme.. PAG. 21

propuse pentru REZOLVARE
***
Probleme de la concursul ACM și de la antrenamentul lotului național.

focus.. PAG. 19

Aritmetică cu numere mari
O singură temă și două abordări. Una aparține lui *TIBERIU SOCACIU* iar cealaltă lui *ION COZAC*.

internet.. PAG. 68

Internet și mass-media
DE DAN IANCU
Companiile de presă atacă Web-ul. Apar însă și alți concurenți.

interviu.. PAG. 32

Despre cărți, școală și informatică
Interviu cu d-na *Irina Athanasiu.*

tehno.. PAG. 36

Memoria virtuală
DE LUCIAN VINȚAN

Sunetul, o necesitate
DE DRAGOȘ MARIAN

portret.. PAG. 46

Povestea mărului
DE GABRIEL PROȘCANU

UNIVAC I
DE GABRIEL PROȘCANU

mate.. PAG. 51

Algoritmi geometrici elementari
DE GRIGORE ALBEANU
Bazele graficii vectoriale se află în geometria analitică.

frontiere.. PAG. 72

Cartea de siliciu
DE MIRCEA SÂRBU
Va rezista oare cartea, așa cum o știm, în Epoca Informaticii?

serial.. PAG. 60

Linux: Sistemul de fișiere
DE CRISTIAN MATEESCU

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.

Culegerea conține:

- probleme de algoritmică, majoritatea rezolvate în Pascal
- probleme de gestiune a bazelor de date (FOX-PRO)
- probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word
- probleme de verificare a cunoștințelor aferente sistemului de operare DOS

Această culegere constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru atestatul de informatică dar poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, culegerea conține o varietate mare de enunțuri de probleme rezolvate, abordând diferite metode și stiluri de programare.

• • •

Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) mai elegant și cu o eficiență în programare mai ridicată.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422] sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Vacanța MARE

Am convingerea că cei mai mulți dintre cititorii noștri sunt acum cu mult mai interesați de șezlonguri decât de calculatoare. Și pentru că destinderea este scopul vacanței, am să încep cu câteva vești bune.

Calculatoare pentru școli

La începutul verii, reprezentanții conducerii companiei engleze ICL - una dintre cele mai mari din Europa - s-au aflat în România, pentru a pune la punct un proiect amplu de informatizare a învățământului românesc. La conferința de presă organizată cu acest prilej, dl. Marian Popa, directorul reprezentanței ICL pentru România și Moldova, ne-a anunțat că proiectul se va desfășura pe parcursul a mai mulți ani și că valoarea sa totală se ridică la circa 350 de milioane de dolari. În mod concret, proiectul se referă la dotarea a 16.000 de unități școlare de nivel pre-universitar cu câte o rețea locală formată din 15 stații de lucru (PC-uri) și un server.

Ministerul Educației Naționale desfășoară la rândul său un proiect asemănător, de mai mică amploare. Din câte am aflat, rețea de comunicații RoEduNet intenționează la rândul ei să-și extindă serviciile, în primul rând din perspectiva capacității de transport, dar și prin prezența directă în toate județele.

Se pare deci că, încetul cu încetul, școlile din România vor putea beneficia din ce în ce mai mult de binefacerile (și necazurile...) aduse de tehnologia modernă. Pentru aceasta însă este nevoie de programe specifice, cunoscute sub denumirea generică de *software educațional*. Am aflat cu satisfacție că, de curând, Ministerul Educației Naționale a emis un ordin cu privire la politica producerii de soft educațional. Printre primele inițiative se numără elaborarea unui catalog cuprinzând programele de acest fel precum și organizarea unei expoziții la începutul lunii octombrie.

Așadar, premisele sunt favorabile. Să sperăm că totul va merge conform planului.

În acest număr

Revenind la "oile noastre", v-am pregătit pentru vară un număr dublu, în care ponderea problemelor rezolvate și a celor propuse este ceva mai redusă. E, totuși, vacanță.

Secțiunea *Focus* din acest număr are o structură mai ciudată: cuprinde două articole cu același titlu! Nu, nu este o greșeală. Este vorba despre două abordări ale aceleiași probleme: aritmetica numerelor (foarte) mari. Scopul este dublu. Pe de-o parte, pornind mai ales de la un articol publicat în numărul 3 ("Un algoritm de factorizare") mulți cititori și-au manifestat interesul pentru acest subiect. Pe de altă parte, vrem să demonstrăm că orice problemă admite abordări diferite și să vă îndemnăm să căutați mereu perspective noi asupra diverselor articole din GInfo. Vom fi bucuroși să le publicăm.

Din nou o secțiune nouă: *Tehno*. Ne propunem ca în cadrul acesteia să vă prezentăm elemente de tehnologie hardware sau software care, fără să fie direct legate de programare, vă dezvăluie aspecte importante ale calculatoarelor moderne. În acest număr un articol despre memoria virtuală (posibilitatea de a utiliza discul hard pentru a crea iluzia unei memorii interne de mari dimensiuni) și unul despre tehnologia sunetului în calculatoare (de la aspecte hard până la softuri pentru muzică). Așteptăm părerea voastră în privința acestui gen de articole.

În fine, începem să vă prezentăm, în *Serial*, sistemul de operare Linux. Vâlva pe care a stârnit-o acest software gratuit este în continuă creștere, apar tot mai multe programe aplicative și este din ce în ce mai folosit în învățământ. Chiar dacă nu folosiți Linux la școală sau acasă, noțiunile pe care vi le prezentăm sunt aplicabile (cu corecțiile necesare) pentru multe sisteme de operare moderne.

Abonamente

Deoarece planul aparițiilor GInfo a fost modificat pe parcurs (motiv pentru care acest număr este dublu) abonamentele pentru primele cinci numere vor acoperi și numărul viitor (septembrie). Așadar, actualii abonați își pot prelungi abonamentele pentru anul școlar 1998-1999 achitând doar contravaloarea a șapte apariții.

Premiul de 200.000 lei

În articolul *Millennium Bug*, publicat în GInfo nr. 3, am oferit un premiu de 200.000 lei pentru "cea mai corectă, eficientă, completă și elegantă implementare a unui unit Pascal cuprinzând semantica datei calendaristice". Câștigătorul premiului este student la Universitatea "Politehnica" din București și se numește **Adrian Șandor**. Felicitări.

Sursa este publicată în situl GInfo (**www.ginfo.ro**). Tot aici veți găsi și o implementare realizată de Florin Sasu pe baza algoritmului prezentat în articolul "Un algoritm de factorizare". Așteptam comentariile voastre.

La toamnă

Începând din 15 septembrie se redeschide lista de discuții *Dialog*. Tot atunci vom publica și regulamentul concursului pentru rezolvitorii de probleme. Pe curând.

Mircea Sârbu

Din GInfo nr. 2

SOLUȚIILE problemelor propuse

După ce ne-am ocupat de problemele date la Olimpiada Națională de Informatică, a venit rândul problemelor propuse în Gazeta de Informatică nr. 2 (martie 1998)

P029801: Ogor

Configurația ogorului se reține în matriceа b, care va fi parcursă de sus în jos, de la stânga la dreapta. La întâlnirea unui tip de plantă (în poziția (i,j)) se determină cea mai mare suprafață de teren drepunghiulară ce are colțul din stânga sus în poziția (i,j) și conține numai un singur fel de plantă (procedura numaratoare).

Odată determinat dreptunghiul, acesta se "elimină" (elementelor lui li se atribuie valoarea șirului de caractere vid: ''), iar operațiile de determinare se continuă în restul matricei.

Listing OGOR.PAS

```
Program Ogor;
  const maxn=50; maxm=50;
  type tipogor=array[1..maxn,1..maxm]
    of string[20];

  var i,j,n,m,buc:Integer;
      k,l,di,dj:Integer;
      f:Text;
      b:tipogor;

  Procedure numaratoare(i,j:Integer;
  var di,dj:Integer);
    Function co_buna(i,j,di,
                     dj:Integer):Boolean;
    var s:Integer;
    begin
      s:=i;
      while (s<=di) and (b[i,j]=b[s,dj])
        do s:=s+1;
      co_buna:=s>di
    end;
begin
    di:=i; dj:=j;
    while (di<n) and (b[di+1,j]=b[i,j])
      do di:=di+1;
    while (dj<m) and co_buna(i,j,di,dj+1)
      do dj:=dj+1
  end;

Begin
  Assign(f,'TEREN.IN');
  Reset(f);
  Readln(f,n,m);
  for i:=1 to n do
    for j:=1 to m do
      Readln(f,b[i,j]);
  Close(f);
  Assign(f,'TEREN.OUT');
  Rewrite(f);
  buc:=0;
  for i:=1 to n do
    for j:=1 to m do
      if b[i,j]<>''
      then
        begin
          buc:=buc+1;
          numaratoare(i,j,di,dj);
          for k:=i to di do
            for l:=j to dj do
              b[k,l]:=''
        end;
  Writeln(f,buc);
Close(f)
End.
```

P029802: Grăniceri

La fiecare pas se citesc din cele N (N<=9) fișiere de intrare câte o înregistrare (care constă în tripletul (zi, ('IN' sau 'OU'), nume).

Despre aceste persoane spunem că trec granița la momentul actual, iar informația despre ele o reținem în vectorul trece. Informația despre toate persoanele care au trecut granița până în ziua actuală (deci care în fișiere se găsesc înainte de înregistrarea curentă) sunt reținute în vectorul trecut. În câmpul sens al acestei variabile se reține sensul ultimei deplasări ('IN' sau 'OU'). Variabila buc contorizează persoanele care au trecut granița fie 'IN' fie 'OU'.

Reamintim că la fiecare pas se citește câte o înregistrare din fișierele de intrare. Pentru fiecare astfel de înregistrare se testează dacă persoana care a făcut deplasarea se află sau nu în vectorul trecut (procedura caut_data_prec). Dacă se află în acest vector atunci se compară sensul deplasării actuale cu sensul ultimei deplasări. Dacă ele coincid înseamnă că persoana respectivă a trecut cel puțin o dată granița ilegal.

La sfârșit, după ce au fost epuizate toate fișierele de intrare, se testează care dintre persoane au sensul ultimei deplasări 'OU', acestea fiind scrise în fișierul 'AFARA.OUT'.

Listing GRANICER.PAS

```
Program Graniceri;
const maxn=1000;
type date=record
           timp:Integer;
           sens:string[2];
           nume:string[40]
         end;
var i,j,n,m,buc:Integer; c,orice:Char;
    f:array [1..9] of Text;
    exista:Boolean; g,h:Text;
    trece:array [1..9] of date;
    trecut:array [1..maxn] of date;

  Function mai_exista_date:Boolean;
  var i:Integer;
  begin
    i:=1;
    while (i<=n) and (trece[i].timp=1000)
      do i:=i+1;
    mai_exista_date:=(i<=n)
  end;

  Procedure cu_data_cea_mai_mica
            (var i:Integer; n:Integer);
  var j:Integer;
  begin
    i:=1;
    for j:=2 to n do
     if trece[i].timp>trece[j].timp then i:=j
  end;

  Procedure caut_data_prec(i,buc:Integer;
          var exista:Boolean; var j:Integer);
  begin
    j:=1;
    while (j<=buc)
      and (trece[i].nume<>trecut[j].nume)
        do j:=j+1;
    exista:=j<=buc
  end;

Begin
  Write('Cate puncte de frontiera exista?');
  Readln(n); Writeln;
  c:='1';
  for i:=1 to n do
  begin
    Assign(f[i],'VAMA.IN'+c);
    Reset(f[i]);
    if Eof(f[i])
    then trece[i].timp:=1000
    else
      with trece[i] do
        Readln(f[i],timp,orice,
                          sens,orice,nume);
    c:=Succ(c)
  end;
  Assign(g,'AFARA.OUT'); Rewrite(g);
  Assign(h,'FRAUDE.OUT'); Rewrite(h);
  buc:=0;
  while mai_exista_date do
  begin
    cu_data_cea_mai_mica(i,n);
    caut_data_prec(i,buc,exista,j);
    if exista
    then
      if trece[i].sens=trecut[j].sens
      then Writeln(h,trece[i].nume)
      else trecut[j]:=trece[i]
    else
      begin
        buc:=buc+1;
        trecut[buc]:=trece[i]
      end;
    if Eof(f[i])
    then trece[i].timp:=1000
    else
     with trece[i] do Readln(f[i],timp,orice,
     sens,orice,nume)
  end;
  for i:=1 to buc do
    if trecut[i].sens='OU'
    then Writeln(g,trecut[i].nume);
  Close(g);
  Close(h);
  for i:=1 to n do Close(f[i])
End.
```

P029803: Camping

Să analizăm în primul rând cazul în care avem o singură căsuță. Problema ar fi echivalentă cu determinarea numărului maxim de segmente dispuse pe o aceeași dreaptă și care nu se intersectează. O primă idee de rezolvare a acestei probleme ar fi metoda *programării dinamice* și anume în modul descris la problema "Închirieri auto".

Totuși, datorită faptului că segmentele au lungimi egale, maximul numărului solicitărilor începând cu ziua `solicit[i]` este dat de segmentul cel mai apropiat de cel curent și care nu se intersectează cu acesta. De aceea nu mai este nevoie de metoda programării dinamice, fiind suficient un algoritm *Greedy*: pentru fiecare zi a anului începând cu prima, determinăm dacă există o solicitare care începe cu acea zi. Dacă da și dacă această căsuță este liberă, satisfacem solicitarea.

În cazul nostru, având `k` căsuțe, aplicăm acest algoritm pentru fiecare căsuță în parte, verificând ca nici unul dintre turiști să nu fie cazat în două căsuțe.

Listing CAMPING.PAS

```
Program Camping;
const MaxN=365; MaxK=100; MaxM=14;
var k,m,n:Integer;
    solicit:array [1..MaxN] of Integer;
            { numarul solicitarilor incepand }
                        { cu ziua solicit[i] }
    rezolvare:Integer;
    cas_libera:array [1..MaxK] of Integer;

  Function Liber(i:Integer):Integer;
  var j:Integer;
  begin
    j:=1;
    while (j<=k) and (cas_libera[j]>i)
      do Inc(j);
    Liber:=j
      { j este prima casuta libera incepand }
                    { cu cea de a i-a zi }
  end;

  Procedure Citire;
  var fi:Text;
      i,x:Integer;
  begin
    Assign(fi,'CAMP.IN');
    Reset(fi);
    Readln(fi,k); Readln(fi,m); Readln(fi,n);
    for i:=1 to MaxN do solicit[i]:=0;
    for i:=1 to N do
    begin
      Readln(fi,x);
      Inc(solicit[x])
    end;
    Close(fi)
  end;

  Procedure Scrie_rezultate;
  var fo:Text;
  begin
    Assign(fo,'CAMP.OUT'); Rewrite(fo);
    Writeln(fo,rezolvare);
    Close(fo)
  end;

  Procedure Calculeaza;
  var i,j:Integer;
  begin
    rezolvare:=0;
    for i:=1 to MaxN do
    begin
      while solicit[i]>0 do
      begin
        j:=Liber(i);
        if j<=K
        then
          begin
            Inc(rezolvare);
            Dec(solicit[i]);
            cas_libera[j]:=i+m
          end
        else Break
      end
    end
  end;

Begin
  Citire;
  Calculeaza;
  Scrie_rezultate
End.
```

P029804: Expediție pe Marte

Aplicăm *metoda programării dinamice*. Pentru fiecare poziție din matrice determinăm dacă se poate ajunge în ea pornind din colțul stânga sus și executând mișcări doar în jos și spre dreapta reținute în matricea `ell`. Un element al matricei are valoarea `true` dacă la el poate ajunge un robot care pleacă din colțul stânga sus, `false` în caz contrar.

Deoarece un robot care întâlnește o stâncă este blocat într-o direcție, este necesar să determinăm din nou pentru fiecare element al matricei dacă se poate ajunge în el plecând din colțul dreapta jos executând mișcări în sus și la stânga. `ell[i,j]=emn[i,j]=true` înseamnă că poziția `(i,j)` poate fi vizitată de un robot care pleacă din poziția `(1,1)` și ajunge în poziția `(m,n)`. Modul de calcul al elementelor matricei `ell` este următorul: `ell[1,1]=true`, deoarece roboții pleacă din această poziție; pentru prima linie și prima coloană avem `true` până la întâlnirea primei stânci, `false` în rest. Pentru orice altă poziție vom avea `ell[i,j]=true` dacă nu există stâncă și dacă cel puțin una din pozițiile `ell[i-1,j]` sau `ell[i,j-1]` este `true`.

Listing MARTE.PAS

```
Program Marte;
const MaxMN=100;
Var t:array[1..MaxMN,1..MaxMN] of 0..2;
    m,n:Word;
    rezolvare:Word;

  Procedure Citire;
  var fi:Text; i,j:Word;
  begin
    Assign(fi,'MARTE.IN'); Reset(fi);
    Readln(fi,m,n);
    for i:=1 to m do
    begin
      for j:=1 to n do Read(fi,t[i,j]);
      Readln(fi)
    end;
    Close(fi)
  end;

  Procedure Scrie_rezultate;
  var fo:Text;
  begin
    Assign(fo,'MARTE.OUT'); Rewrite(fo);
    Writeln(fo,rezolvare); Close(fo)
  end;

  Procedure Calculeaza;
  var i,j:Word;
      e11,emn:array[1..MaxMN,1..MaxMN]
              of Boolean;
  begin
    e11[1,1]:=true;
    for i:=2 to m do
      e11[i,1]:=e11[i-1,1] and (t[i,1]<>2);
    for j:=2 to n do
      e11[1,j]:=e11[1,j-1] and (t[1,j]<>2);
    for i:=2 to m do
      for j:=2 to n do
        e11[i,j]:=(t[i,j]<>2) and (e11[i,j-1]
                              or e11[i-1,j]);
    emn[m,n]:=true;
    for i:=m-1 downto 1 do
      emn[i,n]:=emn[i+1,n] and (t[i,n]<>2);
    for j:=n-1 downto 1 do
      emn[m,j]:=emn[m,j+1] and (t[m,j]<>2);
    for i:=m-1 downto 1 do
      for j:=n-1 downto 1 do
        emn[i,j]:=(t[i,j]<>2) and (emn[i,j+1]
                  or emn[i+1,j]);
    rezolvare:=0;
    for i:=1 to m do
      for j:=1 to n do
      if e11[i,j] and emn[i,j] and (t[i,j]=1)
        then Inc(rezolvare)
  end;

Begin
  Citire;
  Calculeaza;
  Scrie_rezultate
End.
```

P029805: Hotel

Problema, ca grad de dificultate este cea mai simplă dintre aceste probleme propuse, în schimb numărul de linii de cod sursă este cel mai mare în comparație cu cel al celorlalte probleme.

Informațiile despre bucătari și cofetari sunt stocate în doi vectori denumiți sugestiv `bucatar` și `cofetar`, fiecare element al acestor doi vectori fiind o structură de tip înregistrare, având câmpurile: `niv`, `varsta`, `nume`. După ce elementele vectorilor sunt completate cu date din fișierele de intrare corespunzătoare, se trece la analiza întrebărilor. Acestea sunt citite pe rând din fișierul de intrare `INTREB.IN`. Fiecare întrebare este prelucrată cu ajutorul procedurii `desface` și este desfăcută în părțile ei componente și anume: *tipul* solicitat al unui potențial angajat (variabila `ce` va lua valoarea `1` pentru `bucatar`, `2` pentru `cofetar`, `3` pentru `orice`), *relația* (variabila `relatie` va lua valoarea `1` pentru `<`, `2` pentru `>`, `3` pentru `=`, `4` pentru `<=`, `5` pentru `>=`), *numărul* `N` din enunț (în program variabila `nivel`) și *o variabilă booleană* `minmax` care va lua valoarea `true` dacă în câmpul `zzzz` (din enunț) este `MIN`, `false` în caz contrar. Aceste informații fiind determinate se caută (cu procedura `cauta`) în "baza de date" persoana solicitată. În urma executării procedurii, în cazul unei căutări cu succes, în variabila booleană `exista` vom avea `true`, iar numărul întreg `s` va indica poziția în vectorul cu datele de intrare (`bucatar` sau `cofetar`) al înregistrării care satisface condiția din întrebare.

Listing HOTEL.PAS

```
Program Hotel;
const maxn=50;
type date=array[1..maxn] of
           record
             niv:Integer;
             varsta:Integer;
             nume:string[40]
           end;

var i,j,n,m:Integer;
    c:Char;
    f,g:Text;
    bucatar,cofetar:date;
    texte:string;
    ce,relatie,nivel,s1,s2:Integer;
    minmax,este1,este2:Boolean;

  Procedure desface(texte:string;
                    var ce,relatie,nivel:Integer;
                                var minmax:Boolean);
  var i,eroare:Integer;
```

```
begin
  if Copy(texte,1,8)='bucatar:'
  then ce:=1
  else
    if Copy(texte,1,8)='cofetar:'
    then ce:=2
    else ce:=3;
  texte:=Copy(texte,7,Length(texte)-6);
  if ce<>3 then texte:=Copy(texte,3,
                Length(texte)-2);
  case texte[1] of
    '<': if texte[2]='='
         then relatie:=4
         else relatie:=1;
    '>': if texte[2]='='
         then relatie:=5
         else relatie:=2;
    '=': relatie:=3
  end;
  if relatie>3
  then texte:=Copy(texte,3,Length(texte)-2)
  else
    texte:=Copy(texte,2,Length(texte)-1);
  i:=Pos(':',texte);
  Val(Copy(texte,1,i-1),nivel,eroare);
  texte:=Copy(texte,i+1,3);
  minmax:=texte='MIN'
end;

Procedure cauta(n,relatie,nivel:Integer;
                unde:date; minmax:Boolean;
        var exista:Boolean; var s:Integer);
var i:Integer;
  Function mai_mare(i:Integer):Boolean;
  begin
    case relatie of
    1: mai_mare:=unde[i].niv<nivel;
    2: mai_mare:=unde[i].niv>nivel;
    3: mai_mare:=unde[i].niv=nivel;
    4: mai_mare:=unde[i].niv<=nivel;
    5: mai_mare:=unde[i].niv>=nivel
    end
  end;

  Function rel(i,s:Integer):Boolean;
  begin
    if minmax
    then
      rel:=(unde[i].varsta<unde[s].varsta)
      or (unde[i].varsta=unde[s].varsta)
      and (unde[i].nume<=unde[s].nume)
    else
      rel:=(unde[i].varsta>unde[s].varsta)
      or(unde[i].varsta=unde[s].varsta)
      and(unde[i].nume<=unde[s].nume)
  end;
```

```
begin
  s:=1;
  while (s<=n) and not mai_mare(s) do s:=s+1;
  for i:=s+1 to n do
    if mai_mare(i) and rel(i,s) then s:=i;
  exista:=s<=n
end;

Procedure Scrie_rezultate;
begin
  case ce of
  1: if este1
     then Writeln(g,bucatar[s1].nume)
     else Writeln(g,'NU ESTE');
  2: if este2
     then Writeln(g,cofetar[s2].nume)
     else Writeln(g,'NU ESTE');
  3: if este1 and este2
     then
       begin
         if minmax
         then
           if (bucatar[s1].varsta<
                 cofetar[s2].varsta) or
              (bucatar[s1].varsta=
                 cofetar[s2].varsta) and
              (bucatar[s1].nume<=
                 cofetar[s2].nume)
           then Writeln(g,bucatar[s1].nume)
           else Writeln(g,cofetar[s2].nume)
         else
           if (bucatar[s1].varsta>
                 cofetar[s2].varsta) or
              (bucatar[s1].varsta=
                 cofetar[s2].varsta) and
              (bucatar[s1].nume<=
                 cofetar[s2].nume)
           then Writeln(g,bucatar[s1].nume)
           else Writeln(g,cofetar[s2].nume)
       end
     else
       if este1
       then Writeln(g,bucatar[s1].nume)
       else
         if este2
         then
           Writeln(g,cofetar[s2].nume)
         else
           Writeln(g,'NU ESTE')
  end
end;

Begin
  Assign(f,'BUCATAR.IN');
  Reset(f);
  n:=0;
```

```
  while not Eof(f) do
  begin
    n:=n+1;
    Readln(f,bucatar[n].niv,
         bucatar[n].varsta,c,bucatar[n].nume)
  end;
  Close(f);
  Assign(f,'COFETAR.IN'); Reset(f);
  m:=0;
  while not Eof(f) do
  begin
    m:=m+1;
    Readln(f,cofetar[m].niv,
         cofetar[m].varsta,c,cofetar[m].nume)
  end;
  Close(f);
  Assign(f,'INTREB.IN'); Reset(f);
  Assign(g,'RASPUNS.OUT'); Rewrite(g);
  while not Eof(f) do
  begin
    Readln(f,texte);
    desface(texte,ce,relatie,nivel,minmax);
    if ce<>2
    then cauta(n,relatie,nivel,bucatar,
              minmax,este1,s1);
    if ce<>1
    then cauta(m,relatie,nivel,cofetar,
              minmax,este2,s2);
    Scrie_rezultate
  end;
  Close(f); Close(g)
End.
```

P029806: Radar

Sesizările rachetelor sunt citite într-o structură având câmpurile: `timp`, `li` (linie) și `co` (coloană). Pentru fiecare astfel de sesizare se va determina dacă mai există încă două astfel de sesizări care respectă coliniaritatea și viteza cu prima. Dacă sunt determinate, iar baza ar putea fi atinsă de rachetă (funcția `atinge_sol`), soldații vor fi alertați (procedura `alerteaza`). După aceea, pentru a nu se face alertări suplimentare, racheta va fi eliminată din vectorul cu sesizările (`sesiseaza`) prin modificarea valorii câmpului `timp` la 0.

Listing RADAR.PAS

```
Program Radar;
const maxn=1000;
type date=array[1..maxn] of
           record
             timp,li,co:Integer
           end;
var i,n:Integer;
    ses:date; {sesizeaza }
    c:Char;
    f,g:Text;
```

```
Function atinge_sol(i,j,k,tli,tco:Integer;
        var btimp,bli,bco:Integer):Boolean;
  Function se_apropie(li,co:Integer):Boolean;
  begin
    se_apropie:=(li>=496) and (li<=505)
                and (co>=496) and (co<=505)
  end;

  Function afara(li,co:Integer):Boolean;
  begin
    afara:=(li<1) or (li>1000)
                or (co<1) or (co>1000)
  end;

begin
  bli:=ses[i].li;
  bco:=ses[i].co;
  btimp:=ses[i].timp;
  while not se_apropie(bli,bco)
        and not afara(bli,bco) do
  begin
    btimp:=btimp+1;
    bli:=bli+tli;
    bco:=bco+tco
  end;
  atinge_sol:=se_apropie(bli,bco)
end;

Procedure alerteaza(i,j,k,tli,tco:Integer);
var ii,t,btimp,bli,bco:Integer;
begin
 if atinge_sol(i,j,k,tli,tco,btimp,bli,bco)
  then
    begin
      Write(g,ses[k].timp,' ',btimp);
      Writeln(g,' ',bli,' ',bco)
    end;
  for ii:=i+1 to n do
  begin
    t:=ses[ii].timp-ses[i].timp;
    if (t>=9) and
     ((ses[ii].li-ses[i].li) div t=tli) and
     ((ses[ii].co-ses[i].co) div t=tco)
    then ses[ii].timp:=0
  end
end;

Procedure prelucrare(i:Integer);
var j,k,tj,tk,tli,tco:Integer;
    gata:Boolean;
begin
  j:=i+1;
  gata:=false;
  while not gata and (j<=n) do
  begin
    tj:=ses[j].timp-ses[i].timp;
```

```
    if tj>=9
    then
    begin
      if (Abs(ses[j].li-ses[i].li) mod tj=0)
         and
        (Abs(ses[j].co-ses[i].co) mod tj=0)
      then
        begin
          k:=j+1;
          tli:=(ses[j].li-ses[i].li)div tj;
          tco:=(ses[j].co-ses[i].co)div tj;
          while not gata and (k<=n) do
          begin
            tk:=ses[k].timp-ses[j].timp;
            if tk>=9
            then
            begin
              if (Abs(ses[k].li-ses[j].li)
                  mod tk=0) and
                 (Abs(ses[k].co-ses[j].co)
                  mod tk=0) and
                 ((ses[k].li-ses[j].li)
                  div tk=tli) and
                 ((ses[k].co-ses[j].co)
                  div tk=tco)
              then
                begin
                  gata:=true;
                  alerteaza(i,j,k,tli,tco)
                end
            end;
            k:=k+1;
          end
        end
    end;
    Inc(j)
  end
end;

Begin
  Assign(f,'RACHETA.IN');
  Reset(f);
  Assign(g,'RACHETA.OUT');
  Rewrite(g);
  i:=0;
  while not Eof(f) do
  begin
    i:=i+1;
    with ses[i] do Readln(f,timp,li,co)
  end;
  n:=i;
  for i:=1 to n do
    if ses[i].timp>0 then prelucrare(i);
  Close(f); Close(g)
End.
```

P029807: Închirieri auto

Datele de intrare le citim în vectorul `solicit` care reține fiecare solicitare a unui autoturism sub forma unei perechi `(a,b)`, unde `a` reprezintă începutul perioadei de închiriere, iar `b` reprezintă sfârșitul acelei perioade. Rezolvăm problema prin *metoda programării dinamice*. Construim un vector `f` cu `365` de elemente, în care pe poziția `i` reținem numărul maxim de zile (cu număr de ordine mai mic decât `i`) în care a fost închiriat autoturismul. Modul de calcul al elementelor lui `f` este următorul: `f[0]=0`. Apoi

```
f[i]=max{ f[solicit[j].a-1]+solicit[j].
    b - solicit[j].a/solicit[j].b<=i }
```

Semnificația acestei formule este următoarea: dintre toate solicitările care au ziua de expirare înaintea zilei curente (`i`), o alegem pe aceea (să fie aceasta `j`) pentru care suma dintre perioada de închiriere și numărul maxim de zile în care este închiriată mașina înaintea datei de începere a închirierii `j`, să fie maxim.

Listing AUTO.PAS

```
Program Auto;
const MaxN=1000;
      an=365;

var k,m,n,solutie:Integer;
    solicit:array[1..MaxN] of
              record
                a,b:Integer
              end;

  Procedure Citire;
  var fi:Text;
      i,x:Integer;
  begin
    Assign(fi,'AUTO.IN');
    Reset(fi);
    Readln(fi,N);
    for i:=1 to N do
      Readln(fi,solicit[i].a,solicit[i].b);
    Close(fi)
  end;

  Procedure Scrie_rezultate;
  var fo:Text;
  begin
    Assign(fo,'AUTO.OUT');
    Rewrite(fo);
    Writeln(fo,solutie);
    Close(fo)
  end;

  Procedure Calculeaza;
  var f:array[0..an] of Word;
      i,j,max,nou:Integer;
  begin
    f[0]:=0;$$$$
```

```
    for i:=1 to an do
    begin
      max:=f[i-1];
      for j:=1 to n do
        if solicit[j].b=i
        then
          begin
            nou:=f[solicit[j].a-1]+
                 solicit[j].b-solicit[j].a+1;
            if nou>max then max:=nou
          end;
      f[i]:=max
    end;
    solutie:=f[an]
  end;

Begin
  Citire;
  Calculeaza;
  Scrie_rezultate
End.
```

P029808: Robot

Observăm că sunt suficienți n roboți pentru a culege toate piesele. (Această afirmație am dedus-o din faptul că un robot poate culege piesele de pe o coloană întreagă (și nu numai) deci deoarece matricea are n coloane, sunt suficienți n roboți. Pentru a determina numărul minim de roboți observăm că un robot trebuie să ajungă neapărat pe ultima linie și în plus într-o soluție optimă doi roboți nu ajung în aceeași căsuță de pe ultima linie (coborând de sus în jos). Deci numărul minim de roboți va fi egal cu numărul minim de coloane în ale căror cele mai de jos poziții vor ajunge roboții.

Vom pleca cu câte un robot din ultima (cea mai de jos) căsuță a fiecărei coloane și vom parcurge matricea în sus și la stânga încercând de fiecare dată să eliminăm un număr maxim de piese (elemente având valoarea 1 în matrice). Dacă un robot nu culege nici o piesă, înseamnă că acest robot nefiind util, nu face parte din numărul minim necesar.

Listing ROBOT.PAS

```
Program Robot;
const MaxMN=100;
var t:array[1..MaxMN,1..MaxMN] of 0..1;
    m,n,solutie:Word;

  Procedure Citeste_date;
  var fi:Text;
      i,j:Word;
  begin
    Assign(fi,'ROBOT.IN');
    Reset(fi);
    Readln(fi,m,n);
    for i:=1 to m do
    begin
      for j:=1 to n do Read(fi,t[i,j]);
      Readln(fi)
    end;
    Close(fi)
  end;

  Procedure Scrie_rezultate;
  var fo:Text;
  begin
    Assign(fo,'ROBOT.OUT'); ReWrite(fo);
    Writeln(fo,solutie);
    Close(fo)
  end;

  Procedure Calculeaza;
  var i,ii,j:Word;
      u:array[0..MaxMN] of Word;
  begin
    for i:=2 to m do u[i]:=0;
    u[1]:=1;
    for j:=1 to n do
    begin
      i:=M+1; U[0]:=1;
      while i>0 do
      begin
        repeat
          Dec(i)
        until (i=0) or (t[i,j]=1);
        if i=0 then Break;
        ii:=i;
        while u[ii]=0 do Dec(ii);
        u[ii]:=0; u[i]:=1;
        i:=ii
      end
    end;
    solutie:=0;
    for i:=1 to m do solutie:=solutie+u[i]
  end;

Begin
  Citeste_date;
  Calculeaza;
  Scrie_rezultate
End.
```

P029809: Compilator

Se construiește un vector (numit mem) având dimensiunea de 1024 byte, care reprezintă porțiunea de memorie a compilatorului nostru. Completarea locațiilor de memorie se putea face într-o singură etapă, dacă nu ar fi fost ultimele două tipuri de instrucțiuni. În situația dată sunt necesare două etape. În prima etapă completăm șirul cu rezultatul instrucțiunilor de tip 1, 2 și 3, iar pentru instrucțiunile de tip 4 și 5 rezervăm doar octeții necesari (2 pentru instrucțiunea

4 și 1 pentru instrucțiunea 5). La cea de a doua parcurgere vom completa și locațiile astfel rezervate.

Instrucțiunile de tip 4 și 5 necesită reținerea tuturor etichetelor programului (vectorul et), iar apoi completarea vectorulului mem se realizează în felul următor:

- dacă instrucțiunea curentă este de tipul 4 determinăm poziția în memorie (în vectorul mem) de la care încep instrucțiunile precedate de eticheta dintre < > și scriem rezultatul obținut în vector conform specificației instrucțiunii;
- dacă instrucțiunea este de tipul 5, atunci determinăm poziția din memorie de la care încep instrucțiunile atașate etichetei dintre [ ], și scriem în locul rezervat rezultatului instrucțiunii curente conținutul locației de memorie a cărei poziție am determinat-o.

Listing COMPILAT.PAS

```
Program compilator;
                { Rezolvare de Mihai Scortaru }
                { Liceul de Informatica, Cluj }

type eticheta=record
                  nume:string[7];
                  loc:Word
               end;

var pr:array[1..1024] of string[25];
    mem:array[1..1024] of Byte;
    ne,m,n:Byte;
    et:array[1..1024] of eticheta;
    poz:array[1..1024] of Word;
    f,g:Text;
    p:Word;
    s1:string[25];

  procedure citire;
  var x:Char;
      i:Byte;
  begin
    Assign(f,'PROGRAM.IN');
    Reset(f);
    n:=0;
    while not Seekeof(f) do
    begin
      Inc(n);
      Readln(f,pr[N])
    end;
    Close(f);
    Assign(g,'PROGRAM.OUT');
    ReWrite(g)
  end;

  procedure punenumere;
  var i,x,buc:Byte;
  begin
    x:=0;
    buc:=0;
    i:=1;
    while s1[i]<>'(' do
    begin
      x:=x*10+Ord(s1[i])-48;
      Inc(i)
    end;
    Inc(i);
    while s1[i]<>')' do
    begin
      buc:=buc*10+Ord(s1[i])-48;
      Inc(i)
    end;
    for i:=m+1 to m+buc do
      mem[i]:=x;
    Inc(m,buc)
  end;

  procedure punecar;
  var x:Char;
      i,buc:Byte;
  begin
    x:=s1[2];
    i:=5;
    buc:=0;
    while s1[i]<>')' do
    begin
      buc:=buc*10+Ord(s1[i])-48;
      Inc(i)
    end;
    for i:=m+1 to m+buc do
      mem[i]:=Ord(x);
    Inc(m,buc)
  end;

  procedure punesir;
  var i:Byte;
  begin
    i:=2;
    while s1[i]<>'"' do
    begin
      Inc(m);
      mem[m]:=Ord(s1[i]);
      Inc(i)
    end;
  end;

  procedure puneadr;
  begin
    Inc(m,1)
  end;

  procedure puneval;
  begin
    Inc(m,2)
  end;
```

```
procedure prelucrare1;
var i,j,ap,gh:Byte;
begin
  for i:=1 to n do
  begin
    s1:='';
    ap:=0;
    gh:=0;
    for j:=1 to Length(pr[i]) do
    begin
      if pr[i][j]='''' then Inc(ap);
      if pr[i][j]='"' then Inc(gh);
      if pr[i][j]<>' '
      then s1:=s1+pr[i][j]
      else if Odd(ap) or Odd(gh)
           then s1:=s1+pr[i][j]
    end;
    if Pos(':',s1)<>0
    then
    begin
     Inc(ne);
     et[ne].loc:=m+1;
     et[ne].nume:=Copy(s1,1,Pos(':',s1)-1);
     Delete(s1,1,Pos(':',s1))
   end;
 pr[i]:=s1;
 poz[i]:=m+1;
 case s1[1] of
   '0'..'9':punenumere;
   '''':punecar;
   '"':punesir;
   '[':puneadr;
   '<':puneval
  end
 end
end;

procedure puneadr2;
var etc:string;
    i:Byte;
begin
  i:=2;
  etc:='';
  while s1[i]<>']' do
  begin
    etc:=etc+s1[i]; Inc(i)
  end;
  for i:=1 to ne do
    if et[i].nume=etc
    then
      begin
        mem[p]:=et[i].loc mod 256;
        mem[p+1]:=et[i].loc div 256;
        Break
      end
end;
```

```
procedure puneval2;
var etc:string;
    i:Byte;
begin
  i:=2;
  etc:='';
  while s1[i]<>']' do
  begin
    etc:=etc+s1[i];
    Inc(i)
  end;
  for i:=1 to ne do
    if et[i].nume=etc
    then
      begin
        mem[p]:=mem[et[i].loc];
        Break
      end
end;

procedure prelucrare2;
var i:Byte;
begin
  for i:=1 to n do
  begin
    s1:=pr[i];
    p:=poz[i];
    if pr[i,1]='<' then puneadr2
  end;
  for i:=1 to n do
  begin
    s1:=pr[i];
    p:=poz[i];
    if pr[i,1]='[' then puneval2
  end
end;

procedure scrie_rezultate;
var i:Word;
begin
  for i:=1 to m do
  begin
    Write(g,mem[i],' ');
    Write(mem[i],' ')
  end;
  Close(g)
end;

Begin
  citire;
  ne:=0;
  prelucrare1;
  prelucrare2;
  scrie_rezultate
End.
```

Încercați-vă puterile

PROBLEME propuse pentru REZOLVARE

În perioada 14-30 aprilie s-a desfășurat la Liceul de Informatică din Cluj tabăra de antrenament a lotului lărgit (informatică) având ca scop pregătirea și selecționarea echipelor participante la CEOI'98 și Balcaniada de informatică.

P049801: Funcții surjective

problemă propusă de prof. Doina Rancea, Liceul "Tiberiu Popoviciu", Cluj-Napoca

Enunț:

Să se determine numărul funcțiilor surjective `f:A→B`, unde `A` și `B` sunt două mulțimi formate din litere mici ale alfabetului.

Date de intrare:

Pe prima linie a fișierului de intrare `FUNCTII.IN` sunt scrise literele aparținând mulțimii `A`, iar pe a doua linie literele aparținând mulțimii `B`.

Date de ieșire:

În fișierul de ieșire `FUNCTII.OUT` se va scrie numărul funcțiilor surjective `f:A→B`.

Exemplu:

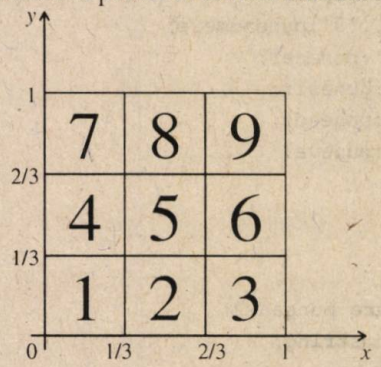
FUNCTII.IN	FUNCTII.OUT
ab	1
x	

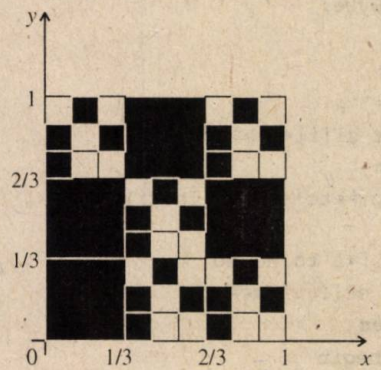
P049802: Fractal

problemă propusă de Cătălin Frâncu, student, Universitatea Politehnică, București

Enunț:

Se consideră în plan mulțimea $A_0=[0,1)\times[0,1)$. Remarcați că A_0 este un pătrat de latură 1 care își conține latura din stânga și pe cea de jos, dar nu și le conține pe cea din dreapta și pe cea de sus. Mulțimea A_0 se împarte în 3×3 pătrate egale de latură 1/3. Pătratele formate au aceeași proprietate cu A_0, adică își conțin laturile din stânga și de jos, dar nu și pe cea din dreapta și pe cea de sus. Numerotăm aceste pătrate în ordinea indicată de figura 1:





■ Pătrate care nu fac parte din mulțimea A_2

□ Pătrate care fac parte din mulțimea A_2

Așadar $P_1=[0,1/3)\times[0,1/3)$, $P_2=[1/3,2/3)\times[0,1/3)$ etc. Din mulțimea A_0 se elimină un set prestabilit de pătrate

(indicat printr-o submulțime a mulțimii {1,2,3,4,5,6,7,8,9,10}), obținând astfel o mulțime A_1. Pentru fiecare pătrat de latură 1/3 care a rămas în mulțimea A_1, se aplică același procedeu: se împarte pătratul în 3×3 pătrățele cu latura de trei ori mai mică (1/9); aceste pătrățele se numerotează în aceeași ordine, apoi **același** set prestabilit de pătrate se elimină. În felul acesta se obține mulțimea A_2. Figura 2 prezintă mulțimea A_2 în cazul în care setul de pătrate eliminate la fiecare pas este {1,4,6,8,10}. Operația de împărțire și eliminare se face **la infinit**, păstrând însă mereu același set de indici de pătrate eliminate. Notăm mulțimea rămasă după repetarea de o infinitate de ori a eliminării cu A_∞. Observați că mulțimea A_∞ nu este vidă. De exemplu, dacă pătratul numerotat cu 1 nu se află în setul de pătrate eliminate la fiecare pas, atunci punctul de coordonate (0,0) se află în mulțimea A_∞.

Se dau patru numere naturale, P_X, Q_X, P_Y, Q_Y, astfel încât $0 \le P_X < Q_X \le 1000$ și $0 \le P_Y < Q_Y \le 1000$. Aceste numere reprezintă un punct de coordonate raționale din mulțimea A_0, $M(P_X/Q_X, P_Y/Q_Y)$. Să se determine dacă punctul M aparține sau nu mulțimii A_∞.

Date de intrare:
Fișierul text FRACTAL.IN conține cel mult cinci seturi de date de test de forma:

P_X Q_X	// Coordonata X a punctului M
P_Y Q_Y	// Coordonata Y a punctului M
NE	// Numărul pătratelor eliminate (0<NE<9)
E_1 E_2 ... E_{NE}	// Indicii pătratelor eliminate (indicii sunt în ordine strict crescătoare)

Date de ieșire: Pentru fiecare set de date de intrare se va afișa pe ecran o linie conținând mesajul 'DA' sau 'NU', după cum punctul M aparține sau nu mulțimii A_∞. Pe ecran nu se vor afișa alte date de ieșire!

Exemplu:

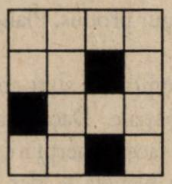
FRACTAL.IN	Răspunsul pe ecran:
1 2	DA
1 2	NU
8	
1 2 3 4 6 7 8 9	
1 3	
2 3	
1	
1	

P049803: Cuvinte încrucișate

Problemă propusă de prof. Ion Maxim, Liceul de Informatică, Suceava

Enunț:
Într-un careu pătratic de cuvinte încrucișate de dimensiune n sunt plasate p puncte negre.
Din fișierul de intrare CAREU.IN se citesc pozițiile (linii) ale celor p puncte negre și cuvinte de lungime cel mult n formate numai din literele mari ale alfabetului englez. Se cere să se completeze careul astfel încât toate cuvintele date să se regăsească în careu (pe orizontală sau pe verticală), eventual separate prin puncte negre.

Într-un careu, numerotarea liniilor și coloanelor se face de la 1 la n începând din colțul din stânga-sus, în jos și spre dreapta.



Date de intrare:
Pe prima linie a fișierului de intrare CAREU.IN sunt scrise cele două numere naturale n (3≤n≤10) și p (0≤p≤25), despărțite printr-un spațiu. Pe următoarele p linii sunt scrise perechi de numere naturale, reprezentând pozițiile punctelor negre. În următoarele linii ale fișierului sunt cuvintele care trebuie plasate în careu.

Date de ieșire:
Fișierul de ieșire CAREU.OUT va conține n linii, fiecare conținând n caractere, reprezentând careul completat. Punctele negre se reprezintă prin caracterul '@'. În calculul lungimii unei linii sau coloane se includ și punctele negre, dacă acestea există pe acea linie sau coloană. Între caracterele din fișierul de ieșire nu există spații.

Exemplu:

CAREU.IN	CAREU.OUT
4 2	C@RA
1 2	ANOD
4 3	SALA
AS	AS@S
ROL	
C	
RA	
S	
CASA	
SALA	
ADAS	
NAS	
ANOD	

P049804: Comerț interplanetar

Problemă propusă de prof. Maria Niță, Liceul "Emanoil Gojdu", Oradea

Enunț:
Datorită revoluției științei și tehnicii, în anul 8991, a devenit posibil comerțul dintre Pământ și cele mai îndepărtate galaxii din Univers. Pământul dorește să

importe produse din câteva din galaxiile sectorului Napoca. Planetele cuprinse în aceste galaxii exportă produse valoroase, de care omenirea are nevoie. Rapoartele preliminare au evidențiat următoarele date:

- numărul de planete din fiecare galaxie este cuprins între unu și 26. Planetele sunt identificate printr-o majusculă.
- fiecare planetă este specializată în producerea și exportul unui singur produs. Planete distincte exportă produse distincte.
- unele perechi de planete sunt conectate prin linii de transport hiperspațiale. Dacă planetele `A` și `B` sunt conectate, ele pot face comerț fără taxe. Dacă planeta `C` este conectată cu planeta `B`, dar nu și cu planeta `A`, atunci `A` și `C` pot face comerț prin intermediul lui `B`, dar `B` își va reține 5% din întregul transport ca și taxă. (Deci `A` va primi doar 95% din ce a trimis `C`, iar `C` va primi doar 95% din ce a trimis `A`). În general, oricare două planete pot face comerț, cu condiția de a fi conectate printr-un set de linii de transport, dar fiecare planetă intermediară de pe parcursul drumului își va reține 5% din întregul transport (ceea ce nu este neapărat echivalent cu 5% din cantitatea originală de produs).
- cel puțin o planetă dintr-o galaxie e dispusă să deschidă o linie de transport care să fie conectată cu Pământul. O astfel de linie cu Pământul, este identică cu oricare linie din galaxie, în materie de comerț. De exemplu, dacă o planetă `K` deschide o linie pentru a se conecta cu Pământul, atunci planeta `K` va face comerț fără taxe cu Pământul, sau poate face comerț cu orice planetă conectată cu `K`, cu condiția să plătească taxele planetelor intermediare.

Există o standardizare prin care s-a asociat o valoare relativă (un număr real pozitiv, mai mic decât `10`) produsului de bază al fiecărei planete. Cu cât numărul este mai mare, cu atât este produsul mai valoros. Mai multe produse de valoare pot fi revândute cu un profit mai mare.

Să se determine planeta care are exportul cel mai mare spre Pământ, atunci când taxele sunt luate în considerare.

Date de intrare:

Fișierul de intrare `IMPORT.IN`, conține descrierea uneia sau mai multor galaxii. Fiecare descriere a unei galaxii începe cu o linie care conține un număr întreg `N` ce reprezintă numărul planetelor din galaxie. Următoarele `N` linii conțin descrierea fiecărei planete sub următoarea formă: O literă (pentru identificarea planetei), un spațiu, valoarea relativă a exportului planetei, sub forma `d.dd`, un spațiu, un string care conține litere și/sau caracterul `'*'`; o literă indică o linie de transport către acea planetă, și `'*'` indică o linie de transport către Pământ.

Date de ieșire:

În fișierul de ieșire, `IMPORT.OUT`, pentru fiecare descriere a unei galaxii, trebuie listată o singură linie de forma `"Import din P"`, unde `P` este litera planetei cu cel mai mare export, după ce taxele de transport au fost luate în considerare. (Dacă mai multe planete au aceeași valoare de export, atunci se va lista planeta care are prioritate alfabetică).

Exemplu:

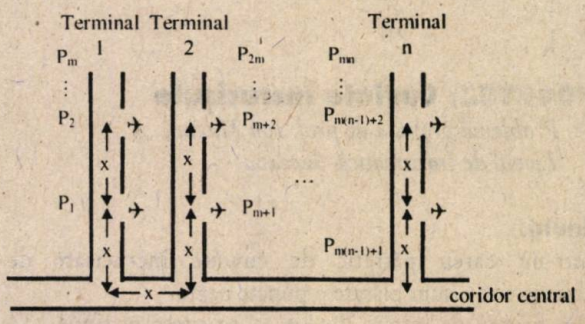
```
IMPORT.IN                IMPORT.OUT
1                        Import din F
F  0.81 *                Import din A
5                        Import din A
E 0.01 *A
D 0.01 A*
C 0.01 *A
A 1.00 EDCB
B 0.01 A*
10
S 2.23 Q*
A 9.76 C
K 5.88 MI
E 7.54 GC
M 5.01 OK
G 7.43 IE
I 6.09 KG
C 8.42 EA
O 4.55 QM
Q 3.21 SO
```

P049805: Aeroport

problemă propusă de asistent Simona Motogna, Universitatea Babeș-Bolyai, Cluj-Napoca și prof. Clara Ionescu, Liceul "Tiberiu Popoviciu", Cluj-Napoca

Enunț:

Pe un aeroport de tranzit sunt `n` terminale și `m` porți pe fiecare terminal. Călătorii aterizează cu un avion și vor decola - în scopul de a ajunge la destinație - cu un alt avion. Aterizarea are loc la un anumit terminal, la o anumită poartă, iar decolarea la altă poartă, eventual de pe alt terminal. În consecință, toți călătorii au de traversat un drum în interiorul aeroportului de la poarta de aterizare la poarta de decolare. Distanțele dintre porți, terminale, respectiv prima poartă de pe terminal la coridorul central, sunt identice (`x=1`).



Realizați o planificare a aterizărilor și decolărilor astfel încât suma lungimilor drumurilor pe care o vor străbate călătorii în aeroport să fie minimă.

Atenție: un avion aterizat nu poate fi programat pentru decolare!

Date de intrare:

Pe prima linie a fișierului AERO.IN se află două numere întregi n și m, reprezentând numărul de terminale, respectiv numărul de porți de pe fiecare terminal (1<=n<=10, 1<=m<=10). Pe următoarele linii se află date cu privire la călătoriile care trebuie planificate sub forma:

loc1 loc2 nr_pasageri

unde loc1 și loc2 sunt șiruri de caractere având cel mult 30 de caractere, reprezentând denumirile localităților, nr_pasageri este un număr întreg (1≤nr_pasageri≤100). Prima localitate reprezintă locul de unde vin călătorii, cea de a doua reprezintă localitatea unde se duc. Datele sunt despărțite printr-un spațiu.

Date de ieșire:

Rezultatele se vor scrie în fișierul text AERO.OUT care va conține pe prima linie un număr întreg, reprezentând suma lungimilor drumurilor pe care o vor parcurge călătorii. Pe liniile care urmează se vor scrie date cu privire la planificarea zborurilor:

localitate poarta

Datele vor fi despărțite prin câte un spațiu.

Exemplu:

AERO.IN
```
2 4
Bucuresti Washington 27
Budapesta New_York 13
Bucuresti Chicago 30
Praga Washington 5
```

AERO.OUT
```
75
Bucuresti 2
Washington 3
Budapesta 5
New_York 6
Chicago 1
Praga 4
```

P049806: Asemănare

problemă propusă de lector Vasile Prejmerean, Universitatea Babeș-Bolyai, Cluj-Napoca

Enunț:

Se dau poligoanele $P=(P_1, P_2, \ldots, P_n)$ și $Q=(Q_1, Q_2, \ldots, Q_n)$ situate în plan, prin coordonatele reale ale vârfurilor succesive $P_i(x_i, y_i)$, și $Qj(x_j, y_j)$, $i,j \in \{1, \ldots, n\}$ ($3 \le n \le 100$).

Se cere sa se determine daca cele doua poligoane P și Q sunt *asemenea*, caz în care se va specifica succesiunea vârfurilor din Q corespunzătoare vârfurilor $P_1, P_2, \ldots, P_n$ din P.

Date de intrare:

Fișierul de intrare POL.IN conține:

n - numărul de vârfuri

x_1 y_1 - coordonatele primului vârf al poligonului P

...

x_n y_n - coordonatele vârfului n al poligonului P

x'_1 , y'_1 - coordonatele primului vârf al poligonului Q

...

x'_n , y'_n - coordonatele vârfului n al poligonului Q

Date de ieșire:

Fișierul de ieșire POL.OUT va avea forma:

i_1 i_2 ... i_n - numerele de ordine ale vârfurilor din Q

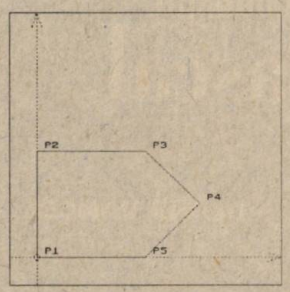
sau

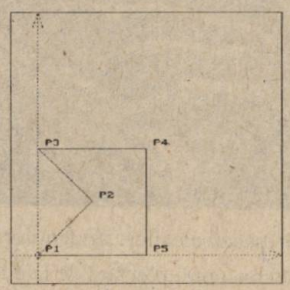
0 - dacă cele două poligoane nu sunt asemenea

Exemple:

POL.IN
```
5
0 0
0 4
4 4
6 2
4 0
0 0
2 2
0 4
4 4
4 0
```

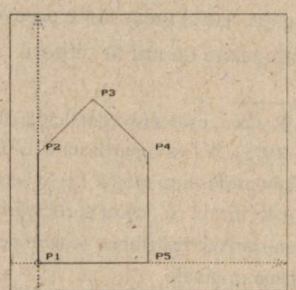
POL.OUT
```
0
```

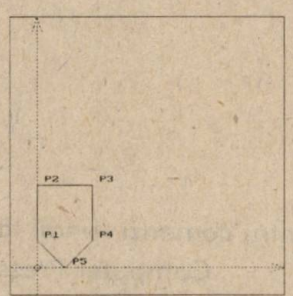




POL.IN
```
5
0 0
0 4
2 6
4 4
4 0
0 1
0 3
2 3
2 1
1 0
```

POL.OUT
```
3 4 5 1 2
sau
2 1 5 4 3
```



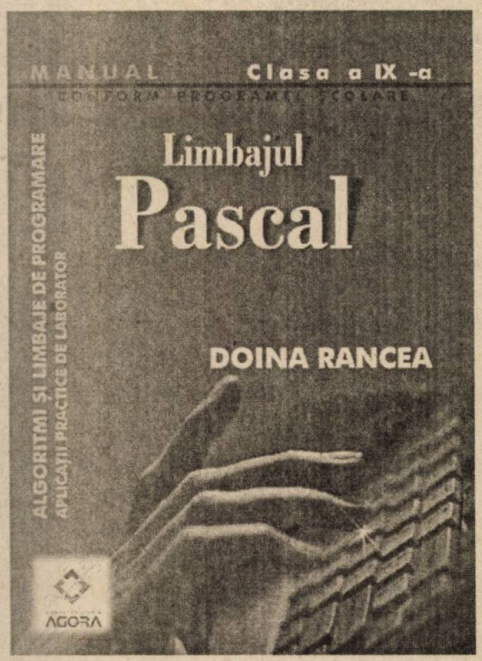


Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie*

... un MANUAL.



Limbajul Pascal este un manual care acoperă programa școlară (1997) a două discipline: „Algoritmi și limbaje de programare" și „Aplicații practice de laborator" care se studiază în clasele a IX-a cu profil de informatică.

Manualul prezintă noțiunile din programă la un nivel accesibil, fiecare noțiune fiind ilustrată în mai multe programe Borland Pascal 7.0. Aceste programe sunt însoțite și de prezentarea datelor de intrare și a rezultatelor în forma sub care apar pe ecran. Problemele au fost selectate în așa fel încât să ilustreze multiplele domenii de utilizare a calculatoarelor. Nu lipsesc problemele de simulare, jocuri, probleme distractive, dar au fost alese și aplicații din programa de matematică de clasa a IX-a cu scopul exemplificării și fixării unor noțiuni importante din matematică. În manual se insistă asupra felului în care se descoperă rezolvarea unei probleme, nivelul de dificultate a acestora crescând gradat, venind astfel în sprijinul elevilor și profesorilor.
Numeroasele probleme din manual permit utilizarea acestuia nu numai pentru studiu individual ci și ca bogat material didactic pentru lecțiile de programare și laborator.
Manualul se adresează elevilor, studenților, profesorilor și oricărei persoane interesate de programare.
Manualul a fost conceput, plecând de la ediția din 1993 (Limbajul Turbo Pascal, clasa a X-a) care a fost regândit substanțial și îmbogățit, astfel încât să fie accesibil unui elev de clasa a IX-a. Ediția anterioară s-a bucurat de un succes deosebit, fiind apreciată de mii de cititori.

Autoarea, Doina Rancea, este absolventă cu diplomă de merit a Facultății de Matematică, Secția Informatică a Universității Babeș-Bolyai din Cluj Napoca (1975). A fost repartizată la Liceul de Informatică din Cluj Napoca unde predă și în prezent limbajele de programare Pascal și C++. Este profesoară gradul I și în această calitate îndrumă activitatea de practică pedagogică a studenților. Experiența bogată a autoarei a fost valorificată cu ocazia participării, în cadrul Comisiei Naționale de Specialitate Informatică a Ministerului Educației Naționale, la actualizări ale programei școlare pentru disciplina informatică și în cadrul comisiilor județene și naționale ale olimpiadelor, propunând probleme originale.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

1, 2, 3, ..., 1E100000

ARITMETICĂ cu numere MARI

Tiberiu Socaciu, Cluj

Încă din clasa întâi copilul ia cunoștință cu numerele. La început îi ajung cele zece degete de la mâini, pentru ca la absolvirea clasei întâi să poată deja manevra numere de ordinul miilor. Probabil că orice absolvent a cinci clase ar trebui să știe să opereze cu numere mult mai mari, chiar dacă nu "știe" să citească numere având peste 12-15 cifre. Dar acesta nu este un lucru care să descurajeze omenirea: numerele cu multe cifre au o largă aplicabilitate în știință și tehnică; și să nu uităm inflația...

Prezentăm în continuare câteva considerații legate de experiența proprie dobândită la implementarea aritmeticii cu numere mari.

Reprezentarea numerelor naturale

În funcție de precizia dorită propunem trei modalități de reprezentare în memorie a numerelor naturale:

1) *reprezentarea în virgulă fixă*
2) *reprezentarea pe șir de bytes*
3) *reprezentarea ca șir de cifre*

1) Reprezentarea în virgulă fixă

Este cea mai simplă modalitate de reprezentare a datelor întregi. Pentru aceasta se va folosi *tipul întreg* (eventual *dublă precizie*) de pe sistemul de calcul. Uzual dimensiunile de reprezentare sunt 2 sau 4 bytes, ceea ce ne asigură o plajă a valorilor distincte ce se pot reprezenta de 2^{15}, respectiv 2^{31} valori pozitive (de obicei semnul se reprezintă pe un bit). Probabil că aceste plaje de valori pot mulțumi orice utilizator obișnuit.

2) Reprezentarea pe șir de bytes

Reprezentarea pe șir de bytes este de fapt o generalizare a reprezentării în virgulă fixă pe un număr de N bytes, adică pe un număr de $8*N$ biți, ceea ce ne asigură că se pot reprezenta 2^{8*N} valori distincte. În acest caz trebuie specificat care dintre cei N bytes este cel mai semnificativ și care dintre cei 8 biți ai unui byte este cel mai semnificativ (uzual cel mai din stânga).

3) Reprezentarea ca șir de cifre

Un număr natural $X=u_1u_2...u_n$ dat în reprezentarea în baza b ($b\geq 2$) și având cifrele u_i din mulțimea $\{0, ..., b-1\}$ ($i=1, ..., n$), nu poate fi codificat în memoria calculatorului dacă n este mare (vezi paragrafele anterioare). Deci se impune memorarea lui X sub forma unui triplet de forma (b, n, L) unde:

- b este *baza sistemului de numerație*
- n este *lungimea* numărului X (numărul de cifre din X)
- L este o *listă liniară* ale cărei elemente sunt *cifrele* lui X

Reprezentarea listei L se va face într-o *zonă contiguă de memorie* de dimensiune m bytes, astfel încât $n \leq m$. Prin urmare fiecare cifră se va memora pe un byte; evident numărul maxim care se poate reprezenta într-o listă de dimensiune m este $N_{max}(m, b)=b_m-1$.

Operații cu numere naturale

În cele ce urmează vom da algoritmii pentru principalele operații cu numere naturale (adunare, scădere, înmulțire, împărțire, comparare etc.) pentru cele trei metode de reprezentare.

1) Operații cu numere naturale reprezentate în virgulă fixă

Acesta este cel mai simplu caz, toate operațiile traducându-se direct în instrucțiuni ale limbajului suport, care la rândul lor sunt traduse în cel mult câteva instrucțiuni mașină adresate direct unității de calcul aritmetice și logice (ALU) din microprocesor (CPU).

2) Operații cu numere naturale reprezentate pe șir de bytes

Algoritmii prezentați aici sunt foarte eficienți, majoritatea lor putând fi implementați direct în limbaj de asamblare. Peste tot n semnifică numărul total de biți (evident el este multiplu de 8), iar a_i, b_i ... sunt cifrele binare de pe pozițiile i ale numerelor A, B etc. Algoritmii de înmulțire și de împărțire sunt preluați din [Szto], cu mențiunea că pentru

înmulțire se poate implementa algoritmul *a la russe* prezentat și în [AnGa].

Algoritmul de adunare: Algoritmul de adunare este foarte simplu: se adună cifrele de același rang cu transportul de la adunarea de rang mai mic și se obține o nouă cifră a sumei, precum și transportul pentru următoarea adunare de rang mai mare. De remarcat că la prima adunare (cea corespunzătoare rangului unităților) transportul anterior (care în mod evident nu există, aceasta fiind prima adunare) trebuie inițializat cu 0 pentru omogenitatea buclei **for**.

Merită atenție specială cazul în care ultimul transport este diferit de 0. În acest caz apare fenomenul numit uzual *depășire* sau *overflow*.

Algoritmul de scădere: Algoritmul de scădere este evident asemănător celui de adunare. Cel mai dificil caz este cel în care se încearcă la scăderea de un anumit rang dintr-o cifră mai mică a unei cifre mai mari. Acest caz trebuie tratat asemănător cu *transportul* de cifră de la adunare. Acest lucru elevii îl numesc *împrumut de la cifra de rang mai mare*. Există și aici posibilitatea unei *subdepășiri* sau *underflow*.

Algoritmul de înmulțire: Pentru înmulțire s-a preferat *algoritmul lui Booth* care transformă înmulțirea în adunări și în scăderi. Considerând că avem de înmulțit numerele

$$X = \sum_{i=1}^{n} x_i \qquad \text{și} \qquad Y = \sum_{i=0}^{n} y_i$$

se poate scrie

$$2X - X = X = \sum_{i=1}^{n} (x_{i+1} - x_i)\,2^{-i} = \sum_{i=1}^{n} \Delta x_i\, 2^{-i}$$

unde $x_{n+1}=0$, în final obținându-se

$$P = XY = \sum_{i=1}^{n} Y \Delta x_i\, 2^{-i}$$

unde diferențele sunt în mulțimea {-1, 0, 1}, ceea ce determină să se adune sau să se scadă (eventual decalat cu un număr de biți) numărul *Y*.

Algoritmul de împărțire: Pentru împărțire prezentăm un algoritm foarte frumos, care este de regulă implementat hard în circuitele specializate pe aritmetică binară. Trebuie neapărat remarcată complexitatea liniară a acestui algoritm!

Intrare: *Numerele* X=X$_1$...X$_n$, Y=Y$_1$...Y$_n$
Pasul 1: Y:=Y*2n
Pasul 2: R:=X
Pasul 3:

```
┌pentru i:=0,n execută
│  R:=R-Y
│  ┌dacă R>=0
│  │ atunci
│  │  Qi:=0
│  │  R:=R+Y
│  └■
│  altfel
│    Qi:=1
└■
```

Pasul 4: R:=R*2^{-n}
Ieșire: R,Q

Algoritmul de comparare: Evident că cea mai simplă modalitate de a compara două numere este compararea cifrelor de același rang de la stânga spre dreapta. În caz de egalitate se trece la compararea cifrelor de rang imediat mai mic. Dacă pentru un anumit rang am găsit că cifra curentă a unuia dintre numere este mai mare decât cifra curentă a celuilalt număr, putem spune că și între numere există aceeași relație. În cazul în care până la epuizarea cifrelor nu apare neegalitate, atunci putem spune că cele două numere sunt egale.

Alte operații: Celelalte operații care sunt implementate nu folosesc structura internă a datelor, ele apelând la operațiile primitive definite anterior. Aceste operații sunt *calculul celui mai mare divizor comun* și *calculul celui mai mic multiplu comun* (care se poate obține din cel mai mare divizor comun) și operațiile de *conversie dintr-o bază în alta*, mascate sub forma *operațiilor de intrare-ieșire*.

3) Operații cu numere naturale reprezentate ca șir de cifre

Au fost implementați algoritmii de adunare, scădere, înmulțire dați în [Knuth], iar pentru cel de împărțire a fost implementată îmbunătățirea dată în [Scala] a algoritmului similar dat în [Knuth]. De asemenea au fost implementate rutine de *calcul al celui mai mare divizor comun* și *de comparare*.

Algoritmii de adunare și scădere: Este evident că algoritmii de adunare și de scădere sunt analogi cu cei prezentați în cazul reprezentării pe biți, singurele diferențe fiind:

- baza de numerație poate să difere;
- nu mai apar operațiile de depășire și subdepășire, lungimea reprezentării operanzilor și a rezultatului fiind dinamică în timp (nu se rețin cifrele nule din fața numărului).

Algoritmul de înmulțire: Algoritmul de înmulțire cel mai simplu este bazat pe înmulțiri polinomiale. Astfel, fiecărui număr i se atașează un polinom ai cărui coeficienți sunt cifrele numărului. Evident că fiecare număr este

valoarea polinomului atașat în baza dată. Deci după o înmulțire (ce comportă o complexitate pătratică) și o normalizare a coeficienților (un fel de "transport" ce are o complexitate liniară) se poate obține polinomul atașat rezultatului, de unde se poate obține rezultatul.

Algoritmul de împărțire: Distingem două cazuri:
- împărțitorul are o cifră;
- împărțitorul are mai mult de două cifre.

Am preferat "spargerea" algoritmului în doi algoritmi distincți din cel puțin două motive:
- cei doi algoritmi sunt bine individualizați;
- în caz contrar, algoritmul ar deveni "mai mare", oricum selectarea unuia din cei doi subalgoritmi trebuind făcută cu o instrucțiune de selecție simplă (un **if**).

Cazul împărțitorului cu o cifră: Prezentăm cazul împărțirii cu un număr format dintr-o singură cifră. O privire mai atentă asupra algoritmului ne va arăta că de fapt cele *n* iterații corespund determinării celor *n* cifre, prin clasica metodă de împărțire.

Intrare: A=$a_n...a_0$, B *în baza* X
Pasul 1: i:=1; t:=0
Pasul 2: c_i:=(t*X+a_i) **div** B
Pasul 3: t:=(t*X+a_i) **mod** B
Pasul 4: i:=i+1
Pasul 5:
```
┌dacă i>n
│  atunci Stop
│  altfel Salt la Pasul 2
└■
```
Ieșire: $c_n...c_0$.

Cazul împărțitorului cu mai multe cifre: Prezentăm îmbunătățirea algoritmului din [Knuth] dată în [Scala]:

Intrare: A=$a_{m-n}...a_0$, B=$b_n...b_0$, *în baza* X, b_n<>0, n>1
Pasul 1: d:=X/(b_n+1); A:=A*d
B:=B*d { *Observație: se introduce încă o cifră* a_0 }
Pasul 2: j:=0
Pasul 3:
```
┌dacă aj=bn
│  atunci q':=X-1
│  altfel
│    q':=(aj*x+aj-1) div bn
│    ┌cât timp b2*q'>X*(a2*X+a1
│      -q'*b1)+a0 execută
│    │  q':=q'-1
│    └■
└■
```
Pasul 4: W:=q'*B
Pasul 5: q_j:=q'
```
┌dacă aj...aj-n<W
│  atunci Salt la Pasul 6
│  altfel Salt la Pasul 7
└■
```
Pasul 6:
```
q':=q'-1
W:=q'*B
qj:=q'
```
Pasul 7:
```
aj...aj-n:=aj...aj-n-W
j:=j+1
┌dacă j<=n
│  atunci Salt la Pasul 3
│  altfel Salt la Pasul 8
└■
```
Pasul 8:
```
rest:=0;
┌pentru i:=1,n execută
│  z:=X*rest+a
│  rest:=z mod d
│  ri:=z div d
└■
```
Ieșire: *Câtul* Q=$q_0...q_m$, *restul* R=$r_n...r_0$

Algoritmul de comparare: În esență, algoritmul de comparare este identic cu cel dinainte, cu mențiunea că apare la început un test relativ la numărul de cifre. Dacă unul din numere are mai multe cifre decât celălalt, atunci acel număr este cu siguranță mai mare.

Calculul celui mai mare divizor comun: Cel mai simplu pentru implementare este *algoritmul lui Euclid*:

Intrare: A, B
Pasul 1: R:=A **mod** B
Pasul 2:
```
┌dacă R<>0
│  ┌atunci
│  │  A:=B
│  │  B:=R
│  │  Salt la Pasul 1
│  └■
└■
```
Ieșire: R

Se știe (vezi [Knuth]) că deși este cel mai cunoscut și mai vechi algoritm nu este și cel mai performant. Pentru reprezentarea pe șir de bytes este însă de preferat implementarea algoritmului *GCD binar*. Acest algoritm iterativ încearcă coborârea valorilor absolute pentru care se calculează cel mai mare divizor comun prin folosirea proprietăților de paritate: $(2a, 2b)=2(a, b)$, $(2a, 2b+1)=(a, 2b+1)$, $(2a+1, 2b)=(2a+1, b)$, unde testul de paritate se face de regulă prin testarea ultimului bit dacă este 0, iar trecerea de la un număr par la jumătatea lui se face printr-un *shift la dreapta* a întregii reprezentări cu 1 bit.

Calculul celui mai mic numitor comun: Pentru a calcula cel mai mic multiplu comun se poate folosi cu succes formula: $(a,b)*[a,b]=a*b$.

Conversia dintr-o bază în alta: Conversia dintr-o bază în alta se realizează prin algoritmul clasic de împărțire repetată la noua bază, toate calculele făcându-se

în vechea bază.

Intrare: *A în baza X, baza Y*
Pasul 1: `n:=0`
Pasul 2: *împarte pe A la Y și obține câtul C și restul R*
Pasul 3:
```
┌dacă R=0
│  atunci Stop
│  altfel Salt la Pasul 4
└■
```
Pasul 4:
```
n:=n+1
c_n:=R
A:=C
```
Salt la Pasul 2
Ieșire: $c_n \ldots c_1$

Implementarea: Implementarea a fost făcută în limbajul Pascal și prezentată pe larg în [Socaciu94] și [Socaciu95b]. Cei interesați de obținerea surselor complete sunt rugați să contacteze autorul. Prin efectuarea de teste s-a ajuns la concluzia că raportul timp de calcul/număr maxim de cifre este în favoarea metodei de reprezentare ca șir de bytes, obținându-se un rezultat foarte apropiat de cazul aritmeticii cu întregi în virgulă fixă.

Calculul factorialelor: Programul `FACTORIA.PAS` calculează factorialele numerelor prin înmulțire, iar apoi prin împărțire verifică corectitudinea calculului. Prin verificare a rezultat că 146! are 255 de cifre. Cu titlu de exemplu oferim cele 158 de cifre ale lui 100!:
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000.

Calculul puterilor lui 2: Programul `2LAN.PAS` calculează puterile lui 2 până la un exponent maxim dat făcând doar operații de adunare (pentru înmulțirea cu 2), iar apoi verifică calculele prin împărțiri succesive la 2. Prin verificare a rezultat că 2^{844} are 255 cifre. Cu titlu de exemplu oferim cele 31 de cifre ale lui 2^{100}:
1267650600228229401496703205376.

Conjectura palindroamelor: Vom enunța *Conjectura palindroamelor*, așa cum se regăsește ea în [Gardner], după care vom prezenta programul pentru verificarea punctuală a conjecturii. Spunem că numărul natural: $X=a_1...a_n$ este un *palindrom* dacă $a_i=a_{n-i}$, $i=1, ..., n$, adică cifrele sale sunt în oglindă. Astfel numărul 12321 este un palindrom. Dacă definim funcția *oglindă*:$N \to N$ dată de *oglindă*$(a_1...a_n)=a_n...a_1$, unde $a_1...a_n$ este un număr natural pentru care $a_1 <> 0$, atunci putem spune că palindroamele sunt *punctele fixe* ale aplicației oglindă, adică numărul natural X este palindrom <=> *oglindă*$(X)=X$. Fiind dat numărul natural X_0 vom defini recursiv șirul $(X_n)_{n>=0}$ astfel: $X_{n+1}=X_n+$*oglindă*(X_n). În [Gardner] se enunță următoarea *conjectură* (adică afirmație presupusă adevărată, încă nefiind demonstrat contrarul ei): "oricare ar fi X_0 număr natural, există n natural, astfel încât X_n să fie palindrom", adică oricare ar fi numărul de pornire, după un număr finit de pași se obține un palindrom. După cum se observă, enunțul conjecturii nu depinde de baza sistemului de numerație ales. Se pare că totuși conjectura a fost rezolvată pentru cazul sistemului de numerație binar. Oferim în continuare rezultatul rulării programului `PALINDRO.PAS` pentru numărul 1993:

Pasul 1: 1993+3991=5984
Pasul 2: 5984+4895=10879
Pasul 3: 10879+97801=108680
Pasul 4: 108680+086801=195481
Pasul 5: 195481+184591=380072
Pasul 6: 380072+270083=650155
Pasul 7: 650155+551056=1201211
Pasul 8: 1201211+1121021=2322232

Problema nu este atât de simplă precum pare: există numere care pur și simplu se "prăbușesc" într-un palidrom după câteva iterații, dar există și numere care se lasă foarte greu "dobândite" lăsând impresia, după un anumit număr de pași, că nu ar fi adevărată conjectura. O astfel de bătălie s-a dat pentru numărul 196.

Ce mai putem face?

Răspunsul ar fi: multe. Evident că în funcție de problema practică ce urmează a fi rezolvată se pot "inventa" multe lucruri. Amintim câteva alte aplicații posibile, o parte din ele fiind prezentate și în anexele din [Socaciu96b]:

- aritmetică cu întregi mari
- calculator "științific" întreg cu posibilități de calcul și conversie în diferite baze de numerație.

Extinderi...

Evident că, în funcție de cerințele problemei se pot construi ierarhii de clase care să pornească de la aceste idei. Vă prezentăm câteva extinderi care au fost prezentate și implementate în [Socaciu95], [Socaciu96a], [Socaciu96b]:

- aritmetică cu rationali;
- algebră liniară cu rationali;
- algebră cu complecși întregi.

...spre calculul simbolic

De regulă toate sistemele de calcul simbolic au implementați și așa numiții *algoritmi seminumerici* (cu alte cuvinte au motoare de lucru cu numere mari). Noi am folosit această aritmetică pentru modulul de calcul simbolic cu expresii Poisson care nu sunt altceva decât generalizări ale polinoamelor algebrice și ale polinoamelor trigonometrice (mai exact o sumă algebrică de termeni polinomiali ce pot fi însoțiți de o formă trigonometrică dintr-o combinație liniară întreagă și precedată de un coeficient rațional).

Referințe

[AnGa] Răzvan Andonie, Ilie Gârbacea, *Algoritmi fundamentali. O perspectivă C++*, Ed. Libris, Cluj-Napoca 1995

[Gardner] Martin Gardner, *Ha ha ou la comprehension de la mathematique*, Ed. Belin, Paris.

[Knuth] Donald E. Knuth, *Tratat de programarea calculatoarelor. Algoritmi fundamentali (volumul I). Algoritmi seminumerici (volumul III).*, Editura Tehnică, București 1983.

[Scala] Robert-Michel di Scala, *SYPAC: Un system experimental de calcul formel en Pascal*, Thse de 3-ime cycle, l'Université Scientifique et Médicale de Grenoble.

[Socaciu94] Tiberiu Socaciu, *POISSON - Sistem de calcul simbolic*, Comunicare la Colocviul Național de Informatică INFO-IAȘI JUNIOR '94, IAȘI 7-9 aprilie 1994

[Socaciu95a] Tiberiu Socaciu, *Rezolvarea unor probleme de Mecanică cerească cu ajutorul procesorului POISSON*, Comunicare la Colocviul Național de Informatică INFO-IAșI JUNIOR '94, IAșI 5-7 aprilie 1995

[Socaciu95b] Tiberiu Socaciu, *Calcul simbolic simbolic cu aplicații în Mecanica Cerească*, teză de licență, Universitatea "Babeș-Bolyai" Cluj, Facultatea de Matematică și Informatică, 1995

[Szto] I.Sztojanov și alții, *De la poarta TTL la MICROPROCESOR*, Editura Tehnică, București 1987

Domnul Tiberiu Socaciu este licențiat și master al Facultății de Matematică și Informatică a Universității "Babeș-Bolyai" Cluj-Napoca. Lucrează ca inginer de sistem la S.C. SAR Transilvania Asigurare-Reasigurare S.A. Cluj-Napoca. Poate fi contactat prin email: infodata@theoffice.net

Listing DOI_LA_N.PAS

```
{Modul: Demo la aritmetica cu numere mari      }
{Observatii: Calculul puterilor lui 2          }
{Modificare: 25 octombrie 1994                 }

Program Doi_la_puterea_n;
uses crt,natural;
var p1,p2,c,r:nrnatural;
    n,k:Longint;
    s:string;
    f:Text;
Begin
  Assign(f,paramstr(1)); ReWrite(f);
  Clrscr;
  k:=0;
  p1.init(10,'1'); p2.init(10,'2');
  Write('Pana la ce putere calculez: ');
  Readln(n);
  for k:=1 to n do
  begin
    r.aduna(p1,p1); p1.done;
    p1.copiaza(r); r.done;
    s:=p1.afiseaza(10);
    Writeln(f,'2^',K,'=',s,' si are ',Length(s),'
cifre.' )
  end;
  for k:=n-1 downto 0 do
  begin
    c.cat(p1,p2); r.rest(p1,p2);
    if not r.estezero
    then
    begin
      Write(f,' Erori la impartire sau ');
      Writeln(f,'aritmetica este inconsistenta
      ...');
      Close(f); Halt(1)
    end;
    p1.done; p1.copiaza(c);
    r.done; c.done;
    s:=p1.afiseaza(10);
    Writeln(f,'2^',k,'=',s,' si are ',Length(s),'
    cifre.')
  end;
  p1.done;
  Writeln(f,'Totul e O.K.!');
  Close(f)
End.
```

Listing TEST.PAS

```
{ Modul: Demo la aritmetica cu numere mari     }
{ Obs: Calculul numarului de permutari si de   }
{ aranjamente                                  }
{ Modificare: 25 octombrie 1994                }

Program test;
uses crt,dos,natural;
var p1,p2,produs,c,r:nrnatural;
    s:string;
    m,n:Byte;
    out:Text;
Begin
  Clrscr;
  n:=1;
  p1.init(10,'1');
  Assign(out,paramstr(1)); ReWrite(out);
  Write('Dati numarul pina la care calculez: ');
  Readln(m);
  for n:=1 to m do
  begin
    Str(n,s);
    p2.init(10,s);
    produs.inmulteste(p1,p2);
    p1.done; p2.done;
    p1.copiaza(produs); produs.done;
    s:=p1.afiseaza(10);
```

```
    Writeln(out,N,'!=',s,' si are ',length(s),'
    cifre.')
  end;
  for n:=1 to m do
  begin
    Str(n,s); p2.init(10,s);
    c.cat(p1,p2); r.rest(p1,p2);
    if not r.estezero
    then
    begin
      Write(out,' Erori la impartire sau
      aritmetica ');
      Writeln(out,'este inconsistenta .');
      Close(out); Halt(1)
    end;
    p1.done; p2.done;
    p1.copiaza(c);
    r.done; c.done;
    s:=p1.afiseaza(10);
    Writeln(out,m,'!/',n,'!=',s);
    Writeln(out,' si are ',Length(s),' cifre.')
  end;
  p1.done;
  Writeln(out,'Totul e O.K. !');
  Close(out)
End.
```

Listing CONJECT.PAS

```
{ Modul: Demo la aritmetica cu numere mari        }
{ Observatii: Verifica conjectura palindroamelor  }
{ Modificare: 25 octombrie 1994                   }

Program Conjectura_palindroamelor;
uses crt,natural;
var a1,a2,r:nrnatural;
    s:string;
    f:Text;
    k:Byte;

  function palindrom(x:string):Boolean;
  var ok:Boolean;
      i,l:Byte;
  begin
    ok:=true;
    l:=Length(x);
    for i:=1 to l div 2 do
      ok:=ok and (x[i]=x[l+1-i]);
    palindrom:=ok
  end;

  function reverse(x:string):string;
  var y:string;
      i,l:Byte;
  begin
    l:=Length(x);
    y[0]:=Chr(l);
    for i:=1 to l do y[i]:=x[l+1-i];
    reverse:=y
  end;

Begin
  Clrscr;
  Assign(f,Paramstr(1)); Append(f);
  Write('Dati numarul pentru care verific');
  Write(' conjectura palindroamelor: ' );
  Readln(s);
  k:=0;
  a1.init(10,s);
  while not palindrom(a1.afiseaza(10)) do
  begin
    Inc(k);
    a2.init(10,reverse(a1.afiseaza(10)));
    r.aduna(a1,a2);
    Write(f,'Pasul ',K,': ',a1.afiseaza(10));
    Writeln(f,'+',a2.afiseaza(10),'=',
    r.afiseaza(10));
    a2.done; a1.done;
    a1.copiaza(r);
    r.done
  end;
  Writeln(f,'OK! dupa ',k,' pasi.');
  Close(f)
End.
```

PRO-DT
1 CALCULATOR PENTRU FIECARE
COMPUTERE
286, 386, 486,
IMPRIMANTE
A4 - A3
GARANȚIE TEHNICĂ 12 LUNI
LA CELE MAI BUNE PREȚURI
STR. ARGEȘ NR. 21 - CLUJ-NAPOCA
TEL FAX 064/430.111
e-mail: prodtcom@mail.dntcj.ro

„La început a fost numărul" (Pitagora)

ARITMETICĂ cu numere MARI

de Ion Cozac

Cum se reprezintă numerele în calculator, cum instruim calculatorul să facă operații aritmetice cu numere mari? Iată câteva întrebări la care încearcă să răspundă articolul de față. Aritmetica numerelor mari are o serie de aplicații, dintre care cea mai cunoscută este criptografia. În final se prezintă tehnica RSA de criptare cu chei publice.

Reprezentarea numerelor în calculator

Un număr întreg se reprezintă în calculator în baza 2 pe unul, doi, patru (uneori opt) octeți. Numărul $2100_{(10)}$ se reprezintă astfel: $0000100000110100_{(2)}$ (am folosit reprezentarea pe doi octeți). Reprezentarea numerelor pozitive nu ridică nici un fel de probleme pentru efectuarea operațiilor aritmetice. Domeniul valorilor pozitive care pot fi reprezentate este între 0 și 2^N-1, unde N este numărul de biți folosiți pentru reprezentare (8 pentru un octet, 16 pentru doi octeți ș.a.m.d.).

Cum se reprezintă numerele negative? Se pot folosi mai multe metode, fiecare avantajând o categorie de operații aritmetice și dezavantajând pe celelalte; prezentăm aici doar două metode:

- reprezentarea semnului pe o poziție distinctă; numărul $-2100_{(10)}$ se reprezintă astfel:
  $-0000100000110100_{(2)}$;
- reprezentarea în complement față de doi; un număr negativ m se reprezintă ca fiind rezultatul operației 2^N-|m|; reprezentarea numărului $-2100_{(10)}$ se obține astfel:

```
1000000000000000-
 0000100000110100
------------------
 1111011111001100
```

Prima metodă de reprezentare complică operațiile de adunare și scădere, pe când operațiile de înmulțire și împărțire se pot efectua după aceleași reguli ca și pentru numere fără semn (evident, la final se va ține cont de regula semnelor pentru rezultat). Domeniul valorilor care pot fi reprezentate este între -2^N+1 și 2^N-1; pentru numărul zero avem două reprezentări distincte!

A doua metodă (folosită pe majoritatea calculatoarelor) este avantajoasă pentru operațiile de adunare și scădere, care se efectuează ca și pentru numere fără semn. În plus se poate pune în evidență și depășirea capacității de reprezentare a rezultatului. Se contorizează câte transporturi avem în cifra semn și după ea; dacă avem zero sau două transporturi, rezultatul este corect; dacă avem un transport s-a produs depășirea capacității de reprezentare. Operațiile de înmulțire și împărțire se complică, deoarece mai întâi se determină valorile absolute ale operanzilor, apoi se efectuează operația cerută pe operanzi pozitivi, și în final se corectează (eventual) semnul rezultatului. Domeniul valorilor care pot fi reprezentate este între -2^{N-1} și 2^{N-1}-1.

Reprezentarea numerelor mari

În cele ce urmează prin numere mari înțelegem valori numerice întregi reprezentate pe un număr de biți care depășește capacitatea de prelucrare a unității aritmetice. Notăm cu W capacitatea de memorare (în biți) a unui registru al unității aritmetice (de exemplu 32 pentru microprocesoarele 80386 și mai noi), și cu N numărul de biți folosiți pentru reprezentarea unui număr mare. Considerăm că un număr mare este o valoare numerică reprezentată în baza B=2W iar cifrele numărului sunt memorate în locații succesive de memorie. Cifrele numărului pot fi memorate fie în ordine crescătoare a puterilor bazei B, fie în ordine descrescătoare. În implementarea noastră folosim convenția Intel (ordinea crescătoare); astfel, numărul

$X=x_0 + x_1 B + ... + x_n B^n$

se memorează astfel începând cu adresa A:

$(A)=x_0$
$(A+1)=x_1$
...
$(A+n)=x_n$

Algoritmii prezentați în continuare consideră că operanzii se reprezintă în format fix pe N biți, și rezultatul se reprezintă tot pe N biți. Situația nu se prezintă întotdeauna astfel; de exemplu, înmulțirea poate produce un rezultat care să se reprezinte pe N×2 biți.

Adunarea și scăderea

Operația `S:=X+Y` se efectuează după următorul algoritm:

```
carry:=0;
pentru i de la 0 la n executa
   (s_i, carry):=x_i + y_i + carry;
```

Am presupus că operația de adunare furnizează atât rezultatul (s_i) cât și informația dacă avem sau nu transport după cifra semn (*carry*); această presupunere este valabilă dacă operația este implementată în limbaj de asamblare.

Operația `D:=X - Y` se efectuează după următorul algoritm:

```
carry:=0;
pentru i de la 0 la n executa
   (d_i, carry):=x_i - y_i - carry;
```

Implementarea pe un calculator cu procesor 80x86 folosește instrucțiuni ADC și SBB (adunare cu transport, scădere cu împrumut).

Pentru efectuarea acestor operații cu numere mari este necesară efectuarea unui număr de N/W operații elementare (ale unității aritmetice).

Înmulțirea

Prezentăm mai întâi o tehnică de înmulțire numită *A la russe* pe următorul exemplu:

13 x 5=1101 x 101=1000001=65.

1101	101	impar
(11010)	10	par
110100	1	impar
100001		

Pe prima coloană se efectuează înmulțiri succesive cu doi, echivalente cu deplasarea numărului spre stânga cu o poziție (operațiile de deplasare sunt implementate pe aproape orice calculator digital). Pe a doua coloană se efectuează împărțiri succesive la doi, echivalente cu deplasarea numărului spre dreapta cu o poziție. Se adună doar acele valori din prima coloană care corespund unor valori impare din cea de a doua (o valoare impară în baza doi are ultima cifră binară egală cu 1).

Operația `P:=XxY` se efectuează după următorul algoritm:

```
U:=X; V:=Y;
cit timp (Va0) executa:
   daca impar(V) atunci P:=P + U;
   lshift(U); rshift(V);
```

Acest algoritm consideră că cei doi operanzi sunt valori binare. Numărul de operații elementare efectuate depinde de mărimea celui de al doilea factor, precum și de numărul de cifre binare din reprezentarea acestuia. Numărul maxim de operații elementare este de $(N\times3/W)\times\log_2(V)$. Acest algoritm poate fi îmbunătățit dacă în prealabil scanăm cei doi factori, și alegem pe post de V pe acela care are ordinul de mărime mai mic (cel mai mic număr de cifre semnificative), și în caz de egalitate pe acela care are cel mai mic număr de cifre binare 1; scanarea se face în N×2 pași. O rezolvare mai simplă este de a compara cei doi factori, și de a alege pe post de V pe cel mai mic; comparația se face în cel mult N/W pași. Operația de înmulțire efectuează în cel mai defavorabil caz $(N\times N\times3)/W$ operații elementare.

Tehnica descrisă mai sus are un ordin de complexitate pătratică în cel mai defavorabil caz. În [Livovschi L, Georgescu H - Sinteza și analiza algoritmilor] este prezentat un algoritm care are un ordin de complexitate $O(N\times\log_2(N))$.

A doua tehnică se bazează pe presupunerea că operația elementară de înmulțire (implementată) furnizează un rezultat pe lungime dublă, partea mai semnificativă (*high*) și partea mai puțin semnificativă (*low*), fiecare parte reprezentându-se pe W biți (majoritatea procesoarelor folosesc această implementare). Prezentăm operația de înmulțire simplificată, `P:=X x v`, unde v este o valoare pe care procesorul o poate prelucra direct:

```
p_0:=0;
pentru i de la 0 la n executa:
   (high, low):=x_i x v;
   (p_i, carry):=p_i + low;
   daca (i<n) atunci p_i+1:=high + carry;
```

Această tehnică poate fi folosită pentru conversia unui număr din format ASCII (extern) în format binar.

Exercițiu pentru cititori. Implementați mai mulți algoritmi pentru înmulțirea numerelor mari, și analizați practic (cu cronometrul calculatorului) performanțele fiecăruia, pentru diferite valori ale celor doi factori. Dacă W=32, de la ce valoare a lui N algoritmul descris în lucrarea [Livovschi, Georgescu] devine mai eficient decât ceilalți?

Împărțirea

Operația de împărțire în baza 2 imită împărțirea în baza 10, dar este mult mai simplă. Cifrele corecte ale câtului se "ghicesc" mult mai ușor: ele nu pot fi decât 0 sau 1, și această valoare rezultă în urma unei simple comparații. Operația de împărțire furnizează în același timp câtul și restul împărțirii. Trebuie să facem precizarea că în aritmetică împărțirea întreagă se efectuează după următoarele reguli:

- semnul câtului este dat de regula semnelor;
- restul este întotdeauna pozitiv sau zero și este mai mic decât valoarea absolută a împărțitorului; operația de împărțire este altfel implementată pe microprocesoarele

80x86 (cu scopul de a simplifica circuitele responsabile): restul are semnul deîmpărțitului.

Înainte de a prezenta algoritmul pentru împărțire să facem câteva precizări:

- DM (copia deîmpărțitului) se compune din DMH (high) și DML (low);
- în interiorul ciclului se determină fiecare cifră binară a câtului; rangul acestei cifre este păstrat în variabila `rangcit`.

Și acum algoritmul pentru împărțire: `(C, R):=D / I;`

```
DMH:=0; DML:=D;
C:=0; rangcit:=N;
repeta
   rangcit:=rangcit-1;
   lshift(DM);
   daca (DMH≥I) atunci
      DMH:=DMH-I;
      C (rangcit):=1;
pina cind (rangcit=0);
R:=DMH;
```

Acest algoritm efectuează în cel mai defavorabil caz (N×N×3)/W operații elementare.

Ca și la înmulțire, se poate elabora un algoritm de împărțire simplificată, care implementează operația `(C, r):=D / v`, unde:

- D - deîmpărțitul - este un număr mare;
- v - împărțitorul - este o valoare pe care procesorul o poate prelucra direct;
- C - câtul - este un număr mare;
- r - restul - este o valoare mai mică decât câtul.

Acest algoritm presupune că operația de împărțire elementară (implementată) se execută astfel: deîmpărțitul este reprezentat pe lungime dublă, partea mai semnificativă și partea mai puțin semnificativă, și furnizează la ieșire câtul și restul (majoritatea procesoarelor folosesc această implementare).

```
high:=0;
pentru i de la 0 la n executa
   low:=x_i;
   (c_i, r):=(high, low) / v;
   high:=r;
```

Această tehnică poate fi folosită pentru conversia unui număr din format binar în format ASCII (extern).

Ridicarea la putere

Pentru a calcula X^e folosim reprezentarea lui e în baza 2. Să considerăm calcularea lui X^{11}. Inițial P:=1 și Z:=X. Valorile P și Z evoluează astfel:

P	Z	E	
X	X	1011	impar
X^3	X^2	101	impar
X^3	X^4	10	par
X^{11}	X^8	1	impar

Și acum algoritmul pentru ridicare la putere:

```
P:=1; Z:=X; F:=E;
cit timp (F≠0) executa
   daca impar(F) atunci P:=P × Z;
   Z:=Z × Z; rshift(F);
```

Numărul de înmulțiri (operanzii sunt numere mari!) efectuate este de $\log_2(E)$ + nr-biti-1(E), deci de maximum $2\times\log_2(E)$.

Remarcă. Algoritmul prezentat în această formă prezintă doar un interes teoretic; rezultatul exponențierii este de multe ori prea mare pentru a putea fi reprezentat în calculator. În schimb prezintă un interes practic deosebit (pentru criptografie) efectuarea operației (X^E) mod M. Trebuie să remarcăm faptul că nu este necesară efectuarea operației (X^E), și apoi determinarea restului. Se poate folosi următoarea proprietate:

(X^E) mod M=$((X^k)$ mod M)×$(X^l$ mod M) mod M, unde k+l=E.

Exerciții pentru cititori. Implementați o funcție care calculează (X^E) mod M. Evaluați practic performanțele implementării și reprezentați grafic timpul de răspuns pentru diferite valori ale lui E (valorile X și M rămânând nemodificate). Graficul urmărește alura funcției logaritmice, dar prezintă oscilații în jurul acesteia, deci nu ne indică o funcție strict crescătoare. Care este cauza acestor oscilații? Reprezentați de asemenea și funcția $\log_2(E)$ + nr-biti-1(E); ce observați?

Inversul multiplicativ

Se dau două numere A și N (A<N) relativ prime (adică cmmdc(A, N)=1). Să se determine o valoare X<N astfel încât (A×X) mod N=1.

```
G0:=N; U0:=1; V0:=0;
G1:=A; U1:=0; V1:=1;
cit timp (G1≠0) executa:
   (Y, G2):=G0 / G1;
   (Y citul impartirii, G2 restul)
   U2:=U0-Y × U1;  V2:=V0-Y × V1;
   U0:=U1; U1:=U2; G0:=G1;
   V0:=V1; V1:=V2; G1:=G2;
```

La sfârșit valoarea V0 este inversul multiplicativ căutat. Pentru a putea implementa acest algoritm folosind numai operanzi pozitivi sunt necesare câteva modificări:

- operația `(U2:=U0-Y × U1)` se corectează astfel:

```
Z:=Y  × U1;
cit timp (U0<Z) executa:  U0:=U0 + N;
```

- operația `(V2:=V0-Y × V1)` se corectează în mod analog.

Algoritmul RSA

Se determină mai întâi două numere prime mari diferite, P și Q. Se efectuează:

N:=P × Q; S:=(P-1) × (Q-1);

F este indicatorul lui Euler j(N) și indică câte numere până la N sunt relativ prime cu N.

Se alege un număr E mai mic decât N și relativ prim cu N. Se determină numărul D invers multiplicativ al lui E în raport cu F. Se comunică lumii (se fac publice) valorile N și E, și se ține secretă valoarea D.

Dacă cineva dorește să ne transmită un mesaj M, va proceda astfel:

- se formează din mesajul M grupe de cifre astfel încât o grupă să fie un număr G mai mic decât M;
- se calculează $C=G^E \bmod N$; C este mesajul codificat care se poate transmite.

Decodificarea mesajului se face calculând $C^E \bmod N=G^{DE} \bmod N$; această valoare este chiar G, pe baza teoremei lui Euler - Fermat:

$G^{j(N)} \bmod N=1$ pentru orice număr G relativ prim cu N.

Puterea acestei metode se bazează pe faptul că este foarte dificilă factorizarea (descompunerea în factori primi) a unui număr mare N. Pentru informații suplimentare se poate consulta lucrarea [Victor Patriciu - Criptografia și securitatea rețelelor de calculatoare].

Numerele lui Mersenne

În general este foarte dificil de a determina dacă un număr P foarte mare este prim, și de aceea în majoritatea cazurilor se folosesc algoritmi care furnizează informația: "probabilitatea ca P să fie un număr compus este mai mică decât e", unde e este o valoare precizată. Există unele o categorie specială de numere prime care se pot obține pe baza următoarei teoreme:

2^n-1 este prim dacă și numai dacă n este prim impar și $l_{n-2}=0$ unde: $l_0=4$ și $l_{j+1}=(l_j^2-2) \bmod (2^n-1)$.

Atenție: dacă P și Q sunt numere ale lui Mersenne, atunci produsul P × Q se poate descompune foarte ușor.

Implementarea

Operațiile aritmetice elementare (adunare, scădere, înmulțire, împărțire) cu numere mari fără semn au fost implementate într-un fișier `OPER.C` (pentru Borland C++ 3.1). Prezentăm în continuare interfața cu biblioteca de funcții implementate:

```
#define LGNR 8
#define LGWO 4
#define LGTN LGNR*LGWO
typedef unsigned long TNumar[LGNR]
void aduna(TNumar *s, TNumar *m, TNumar
*n);
void scade(TNumar *d, TNumar *m, TNumar
*n);
void multip(TNumar *p, TNumar *m, TNumar
*n);
void divide(TNumar *c, TNumar *r, TNumar
*m, TNumar *n);
unsigned nenul(Tnumar *n);
int compn(TNumar *m, Tnumar *n);
```

Explicații:

1. un număr mare se reprezintă pe 8 (LGNR) cuvinte duble de memorie (LGWO=4 octeți) (cu alte cuvinte se folosesc 8 cifre în baza 2^{32} pentru reprezentare);

2. declararea unor variabile de tip "număr mare" se face astfel: `Tnumar a,b,c;`

3. apelul unei funcții: `aduna(&a,&b,&c);`

4. se pot declara și punctatori la variabile de tip "număr mare": `Tnumar *pa,*pb,*c;`

5. apelul unei funcții în acest caz: `aduna(pa,pb,pc);`

6. sunt implementate următoarele funcții:
   - adunarea: `S:=M+N;`
   - scăderea: `D:=M-N;`
   - înmulțirea: `P:=M×N;`
   - împărțirea: `(C, R):=M/N;`
   - test dacă numărul N este nenul; rezultatul este 1 dacă numărul este nenul, și 0 în caz contrar;
   - compararea a două numere M și N; rezultatul este -1 dacă M<N, 0 dacă M=N, 1 dacă M>N;

Acest fișier poate fi inclus într-un proiect pentru Borland C (cu modificari minime aceste funcții pot fi folosite și într-un proiect pentru Borland Pascal). Deoarece unele secvențe sunt scrise în asamblare folosind instrucțiuni 80386, pentru compilare din mediul Borland C este necesară setarea opțiunilor de compilare:

- Compiler / Advanced Code Generation / Instruction Set 80386 ;
- Compiler / Code Generation / Compile via Assembler .

Pentru compilare cu BCC (linie de comandă) se folosesc comenzile:

```
BCC-3-S oper.c
TASM oper
```

Ion Cozac este lector la Universitatea "Petru Maior" din Târgu-Mureș și poate fi contactat prin e-mail la adresa cozac@uttgm.ro

Listing XXXXXXXXXXXXXX

```
#include <string.h>
#include <alloc.h>
#include <mem.h>

#define LGNR 8
#define LGWO 4
#define LGTN LGNR*LGWO

#define Alloc(Tip) (Tip *)malloc(sizeof(Tip))

typedef unsigned long TNumar[LGNR];
```

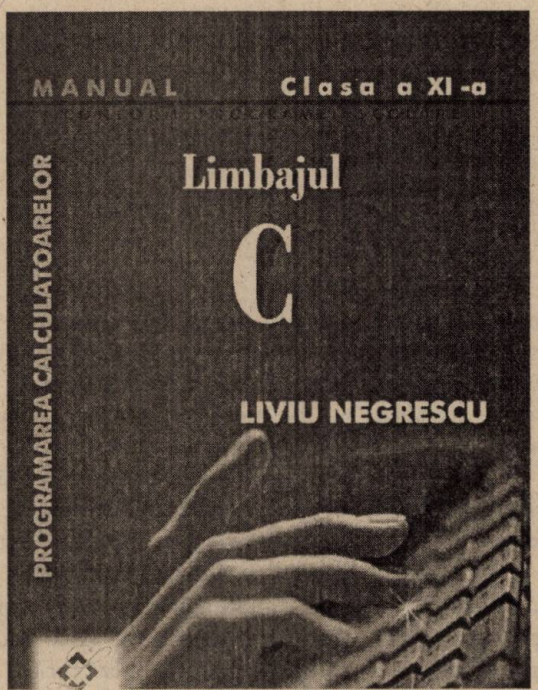
```
static void pascal mshift(void *n, unsigned c)
{
asm {
     les     di,n
     mov     cx,c
     clc
    }
esimprep:
asm {
     mov     eax,es:[di]
     rcl     eax,1
     stosd
     loop    esimprep
    }
}

static void pascal dshift(void *n, unsigned c)
{
asm {
     les     di,n
     mov     cx,c
     clc
    }
esimprep:
asm {
     mov     eax,es:[di]
     rcr     eax,1
     stosd
     loop    esimprep
    }
}

static unsigned pascal impar(TNumar *n)
{
asm {
     les     di,n
     mov     al,es:[di]
     and     ax,1
    }
}

unsigned pascal nenul(TNumar *n)
{
asm {
     les     di,n
     mov     cx,LGNR
     xor     eax,eax
     rep     scasd
     jz      tsemnfin
     inc     ax
    }
tsemnfin:
}

int pascal compn(TNumar *m, TNumar *n)
{
asm {
     std
     mov     bx,ds
     lds     si,m
     add     si,(LGNR-1)*LGWO
     les     di,n
     add     di,(LGNR-1)*LGWO
     mov     cx,LGNR
     repe    cmpsd
     jz      ncompfin
     mov     cx,1
     ja      ncompfin
     neg     cx
    }
ncompfin:
asm {
     mov     ax,cx
     mov     ds,bx
     cld
    }
}

void pascal aduna(TNumar *s, TNumar *m, TNumar *n)
{
asm {
     push    ds
     lds     si,m
     lgs     bx,n
     les     di,s
     mov     cx,LGNR
     clc
    }
adunarep:
asm {
     lodsd
     adc     eax,gs:[bx]
     stosd
     sahf
     add     bx,4
     lahf
     loop    adunarep
     pop     ds
    }
}

void pascal scade(TNumar *d, TNumar *m, TNumar *n)
{
asm {
     push    ds
     lds     si,m
     lgs     bx,n
     les     di,d
     mov     cx,LGNR
     clc
    }
scaderep:
asm {
     lodsd
     sbb     eax,gs:[bx]
     stosd
```

```
        sahf
        add     bx,4
        lahf
        loop    scaderep
        pop     ds
      }
}

void pascal multip(TNumar *p, TNumar *m, TNumar *n)
{
 TNumar *fm,*fn;
 fm = Alloc(TNumar); memcpy(fm,m,LGTN);
 fn = Alloc(TNumar); memcpy(fn,n,LGTN);
 memset(p,0,LGTN);
 while (nenul(fn))
        {
         if (impar(fn)) aduna(p,p,fm);
         mshift(fm,LGNR); dshift(fn,LGNR);
        }
 free(fm); free(fn);
}

void pascal divide(TNumar *c, TNumar *r, TNumar *m,
TNumar *n)
{
 TNumar *dm;
 unsigned long *v,l; unsigned ss;
 dm = (TNumar *)malloc(LGTN*2);
 memset(dm+1,0,LGTN); memcpy(dm,m,LGTN);
 v = (unsigned long *)c+(LGNR-1); l = 0x80000000;
 memset(c,0,LGTN);
 for (ss=LGTN*8; ss>0; ss--)
      {
       mshift(dm,LGNR*2);
       if (compn(dm+1,n)>=0)
          {
           scade(dm+1,dm+1,n); *v |= l;
          }
       l >>= 1;
       if (!l)
          {
           v--; l = 0x80000000;
          }
      }
 memcpy(r,dm+1,LGTN); free(dm);
}
```

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Această carte este concepută în conformitate cu Programa analitică a clasei a XI-a, profilul Informatică. Ea poate fi considerată o introducere elementară în limbajul C. Fiecare noțiune este explicată în mod clar și utilizată în programe simple. De asemenea, capitolele sunt însoțite de exerciții propuse care contribuie la fixarea și înțelegerea noțiunilor întâlnite în capitolele respective.

Cartea se adresează atât elevilor cât și celor interesați să se inițieze în limbajul C care în prezent este unul dintre cele mai utilizate limbaje de programare.

Liviu Negrescu este cercetător principal gr. I la societatea SOFTWARE ITC S.A., filiala Cluj, unde se ocupă cu dezvoltări de programe de sistem și aplicative.
În același timp, desfășoară o activitate didactică prestigioasă de peste 15 ani, la Facultatea de Calculatoare de la Universitatea Tehnică din Cluj.
În decursul anilor a fost titular a mai multe cursuri, ca de exemplu:

- Limbaje de programare și compilatoare;
- Proiectarea translatoarelor etc.

Liviu Negrescu are o serie de lucrări științifice publicate în diferite reviste de specialitate. În colaborare cu alți cercetători de la SOFTWARE ITC S.A. a implementat sisteme cu aplicații în diferite domenii:

- Construcția de compilatoare;
- Interfețe utilizator;
- Prelucrări de date din domenii economice și medicale.

Liviu Negrescu este autorul mai multor cărți apărute în decursul anilor, cu privire la diferite limbaje de programare (CPL1-subset al limbajului PL/1, CDL-limbaj de descriere al compilatoarelor precum și referitoare la limbajele C și C++. Lucrarea beneficiază de experiența vastă a autorului câștigată atât în activitatea de cercetare cât și în cea didactică.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

În dialog cu d-na Irina Athanasiu

Despre CĂRȚI, școală și INFORMATICĂ

de Mircea Sârbu

Cu ocazia Târgului de Carte București 1998 a fost lansată cartea "Limbajul Java - o perspectivă pragmatică", scrisă de d-na Irina Athanasiu împreună cu un grup de colaboratori. Dat fiind interesul pe care cititorii noștri îl manifestă pentru acest limbaj, doamna Irina Athanasiu a avut amabilitatea să ne acorde un interviu.

De ce Java? Și, mai ales, de ce încă o carte despre limbajul Java?

Deoarece tot ce se întâmplă acum în lumea calculatoarelor este pentru Internet, legat de Internet sau accesibil din Internet. Java este limbajul de programare care devine încet, încet, un standard *de-facto* în toate aceste cazuri. Numele Java este asociat acum cu o colecție de tehnologii legate de Internet și Intranet. În străinătate există un număr imens de cărți legate de limbajul Java. În urmă cu câteva luni existau cam 500 de titluri în limba engleză. Presupun că de atunci au mai apărut câteva... În limba română a apărut în 1996 o carte la editura Computer Press Agora (autor Eugen Rotariu) și o serie de traduceri, nu neapărat semnificative.

"Limbajul Java - o perspectivă pragmatică" reprezintă în același timp o introducere în limbajul Java și o colecție de „rețete" gata să fie utilizate de către cei care vor să dezvolte diferite tipuri de aplicații în Java. În general, o carte de „rețete" este ușor perisabilă, pentru că de cele mai multe ori este legată de o versiune de software pentru care „rețetele" respective funcționează optim. Ceea ce am încercat să realizăm în această carte a fost o îmbinare de concepte și soluții. Astfel, chiar dacă o anumită interfață de programare a deve nit depășită prin apariția unei noi versiuni mai performantă sau mai elegantă, conceptele rămân aceleași.

Credeți ca nivelul cărții este accesibil elevilor de liceu?

Nu se poate da un răspuns ferm: da sau nu. Pentru că pregătirea elevilor de liceu este foarte neomogenă. Dacă luăm în considerare faptul că la un moment dat în programa de învățământ pentru liceele de informatică figurează limbajul C și chiar limbajul C++, răspunsul ar trebui să fie da. Dacă luăm în considerare ceea ce știu majoritatea elevilor după parcurgerea disciplinelor respective, aș răspunde cu nu.

De fapt, poate merită să discutăm o problemă mult mai delicată, și anume cea a planului de învățământ pentru disciplinele de informatică în liceu. Mie nu îmi este clar ce urmăresc cei care se ocupă de acest subiect. Pe hârtie și, în majoritatea cazurilor, pe lângă elevi se „consumă" toate cuvintele legate de știința calculatoarelor. Dacă se compară programa de liceu cu cea universitară, pentru multe subiecte se constată că programa din liceu este chiar mai ambițioasă decât cea universitară. Ce înțeleg din asta elevii și care sunt conceptele cu care pot să opereze, asta este altceva.

În mod corespunzător, atunci când vin la facultate mulți absolvenți de licee informatice au impresia că tot ce se spune este familiar, simplu, dar că profesorii sunt chițibușari atunci când nu consideră că un proiect este bine realizat numai pe criteriul "merge", sau când pretind unui student care poate să programeze "orice" stăpânirea unor concepte fundamentale.

De multe ori, studenți care nu au absolvit licee informatice au rezultate mai bune decât cei care au absolvit licee informatice, pentru că nu au de uitat obiceiuri proaste sau noțiuni vagi. Și când vorbesc de rezultate mai bune nu mă refer exclusiv la cele școlare, ci și de rezultate obținute la interviuri de angajare pentru firme reprezentative.

De câte ori am încercat să deschid o discuție pe acest subiect cu persoane care se ocupă de planurile de învățământ din licee, răspunsurile au fost de două tipuri (care reprezintă de fapt variațiuni pe aceeași temă): "Nu ați văzut ce bine se descurcă elevii români la olimpiadele internaționale de informatică " sau "În clasa cu care lucrez, eu am elevi care știu ++ și care s-ar plictisi dacă le-aș face mai puțin".

Carte de vizită

D-na Prof. Dr. Ing. Irina Athanasiu a desfăşurat o bogată carieră didactică în cadrul Facultăţii de Automatizări şi Calculatoare din cadrul Universităţii "Politehnica" din Bucureşti, unde a susţinut cursuri în domeniul limbajelor formale şi limbajelor de programare, proiectarea compilatoarelor, testarea şi evaluarea performanţelor sistemelor de calcul, proiectare logică, programare paralelă şi distribuită etc. A participat la numeroase proiecte de cercetare şi a publicat un mare număr de articole în reviste de specialitate. Este membru USENIX, face parte din comitetului de conducere al GURU (Grupul Utilizatorilor de UNIX din România) şi este colaborator permanent al revistei PC Report.

D-na Irina Athanasiu a publicat până acum 19 cărţi - în calitate de autor sau co-autor - printre care: *Tehnici de programare* (Teora, 1993), *UNIX* (Teora, 1993, 1994), *Iniţiere în Turbo Pascal* (Teora, 1995, 1996, 1997).

După părerea mea, ca şi în cazul olimpiadiştilor de matematică sau fizică, şcoala nu prea are legătură cu rezultatele acestora. Poate doar că nu i-a încurcat, le-a motivat absenţele la orele de română şi le-a mai ridicat mediile la istorie. Majoritatea performerilor sunt fie autodidacţi, fie rezultatul pregătirii făcute cu profesori de specialitate entuziaşti, dar în afara orelor de clasă. În acelaşi timp, genul de pregătire pe care o necesită participarea la un concurs de informatică nu este chiar ceea ce se cere în viaţa adevărată. De cele mai multe ori, o rezolvare "în forţă brută" este soluţia cea mai bună într-un concurs la care contează numărul de probleme rezolvate şi timpul de rezolvare. În viaţa adevărată problemele sunt mai puţin ciudate, în schimb contează aspecte de genul: eficienţa şi claritatea programelor, portabilitatea, reutizabilitate, calitatea documentaţiei, utilizarea unor metodologii coerente de proiectare a programelor şi aşa mai departe.

Dacă ne referim la programa de liceu pentru disciplinele de informatică, cred că ar trebui să fixăm ştacheta undeva mai jos, astfel încât majoritatea elevilor să poată să o sară (pentru că dacă nu pot să o sară, elevii sau chiar profesorii, vor trece pe dedesubt, dar tot vor trece). Să fixăm nişte obiective în legătură cu ce ne aşteptăm să ştie (să poată să facă) un elev "mediu" când devine absolventul unui liceu informatic şi, pornind de aici, să stabilim cum arată planurile de învăţământ. Mie mi se pare că modul în care a funcţionat până acum acest proces, a fost de tipul: "se poartă tehnologia client-server, unde putem să introducem acest subiect în planul de învăţământ+", fără să se cerceteze dacă elevii stăpânesc conceptele pe care se bazează această tehnologie sau nu, sau - mai mult - dacă vor ajunge să o utilizeze sau nu.

După câte ştiu, acum se lucrează la reforma curriculară pentru învăţământul pre-universitar. Sper că cei care sunt implicaţi în această acţiune să aibă curajul să "simplifice" planul de învăţământ, în aşa fel încât acesta să nu mai fie un galop ambiţios peste toate subiectele posibile.

Aţi recomanda Java ca "prim limbaj" de învăţat în şcoală? Altfel spus: cum apreciaţi "valoarea didactică" a limbajului în comparaţie cu Pascal sau C (sau altele)?

Se poate programa "bine" sau "prost" în orice limbaj de programare. Cred că problema este cum se utilizează limbajul respectiv pentru a ilustra ce înseamnă programare. În urmă cu câţiva ani, într-o serie de universităţi americane se utiliza drept prim limbaj de programare un limbaj funcţional, numit *Scheme*. Există mai multe argumente pentru o astfel de soluţie. În primul rând, corespondenţa imediată între conceptele fundamentale şi construcţiile din limbaj, simplitatea sintaxei. Un argument important a fost şi cel referitor la faptul că există şanse foarte reduse ca studenţii să cunoască deja limbajul. Din acest motiv, studenţii căpătau direct obiceiuri bune. Dar - ca în orice limbaj de programare - şi în limbajele de funcţionale se poate programa urât, ineficient etc.

În universităţile româneşti s-a utilizat multă vreme limbajul Pascal ca prim limbaj de programare. Fiind un limbaj care a fost dezvoltat pentru a permite o programare "curată", limbajul are toate datele pentru a fi folosit pentru educaţie. Problema este că în viaţa adevărată limbajul este destul de puţin utilizat, limbajele preferate deocamdată de către industria software fiind C sau C++.

Problema este că cele două limbaje au fost construite în alte scopuri decât cele educative. Să considerăm următorul subiect de test grilă, dat la un examen de licenţă:

```
int *f(void) { int x= 1; return &x;}
```

Care este efectul atribuirii *f() = 1;

a) atribuirea are efecte laterale nepredictabile care pot conduce la erori de execuţie

b) ...

Posibilitatea existenței unui astfel de subiect (pentru care răspunsul corect este a) indică faptul că se consumă prea multă energie pentru a explica și pentru a se înțelege aspectele "umbroase" ale limbajului.

Avantajul limbajului Java din acest punct de vedere este că este un limbaj mult mai curat decât C sau C++, fiind în același timp un limbaj orientat pe obiecte. Consider că Java este foarte bun ca prim limbaj de programare (eventual ca prim limbaj de programare orientat pe obiecte) cu condiția ca să nu se încerce o prezentare exhaustivă a limbajului și, mai ales, a tuturor aplicațiilor sale. Dacă programa analitică a unui curs de inițiere în programare s-ar baza pe *întregul* cuprins al cărții noastre, rezultatele ar fi catastrofale. Aceasta pentru că înțelegerea unor capitole ca cele referitoare la fire de execuție, interfața grafică, aplicații de rețea etc. se bazează pe cunoștințe de sisteme de operare, grafică, rețele, care depășesc cu mult cunoștințele unui elev sau student care se găsește la primul său curs de programare.

Limbajul Java este deja utilizat cu rezultate foarte bune în numeroase universități americane ca prim limbaj de programare.

În rezumat, răspunsul la întrebarea "Ați recomanda Java ca *prim limbaj* de învățat în școală?" este da, dar...

Până nu demult, limbajele de programare erau doar "unelte", mai mult sau mai puțin "expresive", de nivel mai mult sau mai puțin înalt ș.a.m.d. De ce se bucură Java de atâta atenție?

Java este un studiu de caz foarte bun nu numai pentru știința calculatoarelor, dar și pentru marketing. Dacă nu ar fi fost cea de a doua componentă, Java ar fi fost doar încă o încercare de a propune un nou limbaj de programare, așa cum au existat sute sau poate mii de încercări de care nu își mai aduce aminte nimeni. Calitățile intrinseci ale limbajului nu ar fi fost suficiente pentru succesul pe care îl înregistrează limbajul. "Norocul" acestei încercări se bazează pe potrivirea dintre apariția sa și extinderea interesului pentru Internet. Limbajul a fost promovat printr-o campanie de marketing foarte susținută ca fiind "soluția" pentru programarea în Internet. Faptul că toate informațiile necesare pentru utilizarea sa - documentație (de bună calitate), programele pentru dezvoltarea de aplicații (compilator, depanator, generator de documentații, etc.), bibliotecile - au fost de la început accesibile în Internet, a permis ca soluția să fie testată de foarte multă lume. Această testare a condus la apariția unor soluții tehnologice noi, astfel că numeroase soluții tehnologice noi legate de Internet și Intranet au apărut întâi ca soluții în Java.

Este interesant și faptul că Java a fost "încercat" și ca limbaj de programare pentru aplicații care nu au neapărat legătură cu Internetul, în idea potabilității asigurate de către soluția Java. Și rezultatele obținute sunt încurajatoare.

Ca profesor într-o facultate de specialitate, cum apreciați învățământul de specialitate la nivel liceal?

Într-un anumit fel, am răspuns anterior la întrebarea pe care mi-o pui. Am norocul să fiu profesor într-o facultate care adună mulți dintre cei mai buni absolvenți de liceu din România. Din acest motiv probabil că nu pot să fiu obiectivă în ce privește calitatea învățământului liceal. Consider că învățământul liceal trebuie apreciat prin prisma rezultatelor absolvenților "obișnuiți" și nu prin cea a vârfurilor, care s-ar fi descurcat și se descurcă oricum, în special în domenii de tip tehnic pentru care este posibilă o pregătire individuală în afara școlii.

Ceea ce poate merită să fie amintit este modul în care a evoluat în timp învățământul de specialitate la nivel liceal. În ciuda faptului că - în mod evident - cantitatea de cunoștințe în domeniu acumulate în liceu a crescut, nu același lucru aș putea să îl spun despre *calitatea* acestor cunoștințe. A crescut numărul studenților care încă din anul întâi pot să "butoneze" orice, în schimb a scăzut numărul celor care au răbdare să se gândească înainte de a "butona". Evident că evoluția tehnologică a încurajat această abordare, dar mie mi se pare că ceva se pierde. Și că efectul acestei pierderi se va constata mai târziu, când evoluțiile tehnologice vor conduce poate la soluții foarte diferite de cele curente și pentru care pregătirea superficială de acum nu vor permite o adaptare ușoară.

Credeți că performanțele obținute de elevii români la olimpiadele internaționale - mă refer mai ales la cele de informatică - reflectă nivelul învățământului românesc?

Nici vorbă. Elevii performanți (din punct de vedere al olimpiadelor de informatică) nu au practic legătură cu școala. Nu numai că pregătirea pentru olimpiade nu se face la clasă, dar se face în paralel cu ce se întâmplă la clasă. Nu mai știu care este situația acum, dar în urmă cu câțiva ani, performerii erau scutiți - în mod tacit, desigur - de o serie de obligații școlare. În fond, a considera că rezultatele la olimpiadele internaționale de informatică reflectă nivelul învățământului românesc ar fi ca și cum am spune că rezultatele gimnastelor din lotul național reflectă nivelul pregătirii fizice a elevilor din România.

Tot ce se poate spune este că există copiii talentați, motivați destul și care - puși în anumite condiții de pregătire - sunt în stare de performanțe foarte bune.

Cum ați comenta plasarea României pe un loc codaș (locul 33) în Europa, în urma unui sondaj privind calitatea învățământului pre-universitar?

Unele dintre cauzele plasării învățământului pre-universitar pe un loc atât de catastrofal au origini mai vechi. Ani de zile performanța în în-

vățământul pre-universitar era considerată echivalentă cu rezultatele la matematică. Nu erai bun la matematică, nu erai bun la nimic. Iar matematica respectivă nu era accesibilă, în modul în care era prezentată, la un nivel real de înțelegere pentru majoritatea elevilor. Sondajul respectiv s-a bazează nu pe cunoștințe reproduse din manual, ci pe capacitatea de a rezolva probleme, unele din ele cu un pronunțat caracter practic. Ori așa ceva sigur nu se face în învățământul românesc, în orice caz nu în cel pre-universitar.

Există, desigur, și cauze mai noi. Mulți îi blamează pe profesorii care nu își fac datoria, pe elevii care și-au pierdut interesul pentru școală etc. Sunt răspunsuri prea simpliste.

Ceea ce s-a întâmplat la ultimul examen de bacalaureat sau la cel de admitere la liceu spune în fond același lucru ca și sondajul de care aminteai. Faptul că la unele dintre marile licee din țară nu s-au completat locurile și la unele nici nu au existat destui candidați indică o scădere dramatică a nivelului pregătirii acestor elevi. Faptul că un procent semnificativ de elevi nici măcar nu și-au mai pus problema să continue studiile la nivelul liceului, mi se pare foarte grav și cred că reprezintă o pierdere semnificativă pentru viitorul nostru. Toate aceste elemente trebuie să constituie un grav semnal de alarmă pentru toată lumea, inclusiv pentru cei care nu sunt implicați direct în problemele școlii. Pentru că se va pierde rapid, în câțiva ani, unul dintre avantajele semnificative pe care le aveam în privința calității forței de muncă. Dispariția analfabetismului, nivelul relativ ridicat de pregătire intelectuală a unei părți semnificative din populația țării, existența unor specialiști foarte buni în multe domenii - sunt realizări care se vor pierde cu mult mai repede decât au fost câștigate.

Și cred că nu este pur și simplu vorba de bani, de sărăcie. Este ceva care ține de modul în care este privită școala în general (un "consumator de fonduri"), de statutul profesorilor în societate ("încă niște bugetari care trebuie plătiți"). Indiferent de cât de motivați ar putea să fie elevii, dacă profesorii nu sunt la nivelul așteptărilor, orice motivare se stinge. În schimb, un profesor cu har poate să schimbe viața unor elevi. Cred că secretul stă acum - mai mult decât oricând - în calitatea corpului didactic.

Sigur că foarte mulți profesori buni au plecat din școală, că mulți din cei rămași sunt foarte nemulțumiți și își fac orele conform acestor nemulțumiri. Probabil că atitudinea acestora nu se va schimba nici dacă salariile vor fi de două-trei ori mai mari (am citit undeva că, în urma unui sondaj similar cu cel de care mă întrebi, s-a stabilit că profesorii din România sunt printre cei mai nemulțumiți din lume, fără să aibă cele mai proaste condiții din lume). Soluția este să fie atrași în învățământ profesori noi, tineri, care să își dorească acest statut, această meserie. Nu mi s-a părut că cineva încearcă să facă ceva în direcția aceasta. Se discută mult despre reforma învățământului, dar nu este clar cine va face această reformă, pentru că numai niște planuri de învățământ perfecte și niște manuale perfecte nu vor clinti nimic din ce se întâmplă acum în învățământul pre-universitar.

Vă mulțumesc.

Memoria virtuală

de Lucian Vințan

Memoria virtuală reprezintă o tehnică de organizare a memoriei prin intermediul căreia programatorul "vede" un spațiu virtual de adresare foarte mare și care - fără ca programatorul să "simtă" - este mapat în memoria fizic disponibilă.

În mod uzual, spațiul virtual de adrese corespunde suportului disc magnetic, programatorul având iluzia, prin mecanismele de memorie virtuală (MV), că deține practic o memorie de capacitatea discului hard și nu de capacitatea memoriei fizice de tip RAM dinamic (DRAM), care este limitată actualmente la doar 4-64 MB.

În cazul MV, memoria principală este analoagă memoriei cache între CPU (*Central Processing Unit*) și memoria principală, numai că de această dată ea se situează între CPU și discul hard și nu între CPU și memoria principală, ca în cazul cache-urilor. Deci *memoria principală* (MP) se comportă ca un *cache* între CPU și discul hard. Prin mecanismele de MV se mărește probabilitatea ca informația ce se dorește a fi accesată de către CPU din spațiul virtual (disc), să se afle în memoria principală, reducându-se astfel în mod semnificativ timpul de acces de la *10-15* ms (timp acces discuri), la *50-80 ns* (timp acces DRAM-uri) în tehnologiile actuale!

Paginare

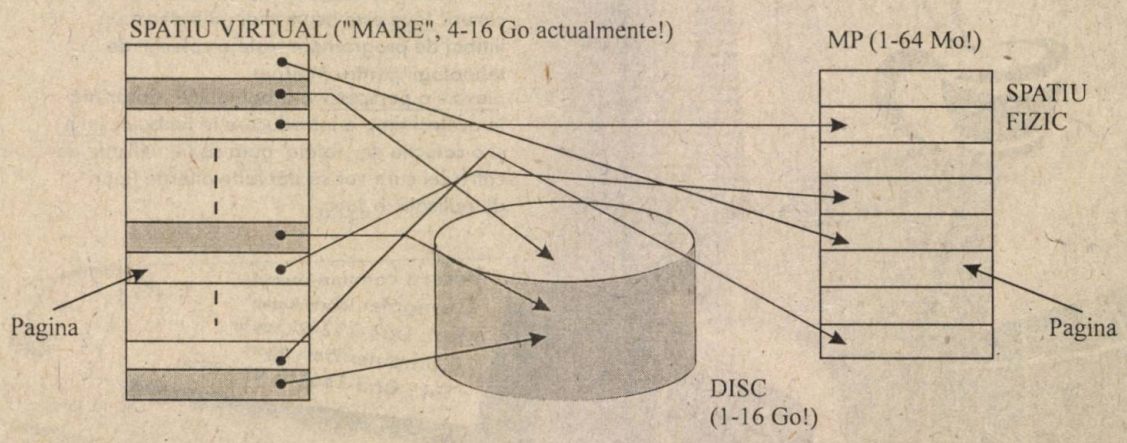
De obicei, spațiul virtual de adresare este împărțit în entități de capacitate fixă (4-64 Ko actualmente), numite *pagini*. O pagină poate fi mapată în memoria principală (oricum este pe disc).

În general, prin mecanismele de MV, memoria principală conține paginile cel mai recent accesate de către un program, ea fiind pe post de "cache" între CPU și discul hard.

Transformarea adresei virtuale emisă de către CPU într-o adresă fizică (existentă în spațiul MP) se numește *mapare* sau *translatare*. MV oferă o funcție de *relocare* a programelor (adreselor de program), pentru că adresele virtuale utilizate de un anumit program sunt mapate spre adrese fizice diferite, înainte ca ele să fie folosite pentru accesarea memoriei. Această mapare permite aceluiași program să fie încărcat în memoria principală, modificările de adrese realizându-se automat prin mapare (fără MV un program depinde de obicei în execuția sa de adresa de memorie unde este încărcat de către sistemul de operare).

MV este un concept deosebit de

Maparea adreselor



util, în special în cadrul sistemelor de calcul ***multiprogramate*** (*multitasking*) care - de exemplu prin tehnici de tip "time-sharing" - permit execuția cvasi-simultană a mai multor programe (cum este cazul în Windows, Unix etc.). Fiecare dintre aceste programe în execuție (procese, task-uri) are propriul său spațiu virtual de cod și date (are alocate un număr de pagini virtuale), dar în realitate toate aceste programe vor ***partaja*** aceeași memorie principală, care va conține, dinamic, pagini aferente diverselor procese. Paginile vor fi dinamic ***încărcate/evacuate*** în/din memoria principală pe disc (spre a permite încărcarea altora, mai "proaspete").

Când o pagină accesată nu se găsește în memoria principală, ea va trebui adusă - prin declanșarea unui mecanism de excepție - de pe disc. Acest proces - analogul *miss*-urilor de la cache-uri - se numește ***"page fault"*** (PF). Evenimentul PF va declanșa un mecanism de excepție care va determina intrarea într-o subrutină de tratare a PF. Aici - prin soft deci - se va aduce de pe disc în memoria principală pagina dorită, după ce în prealabil s-a evacuat eventual o altă pagină. Acest proces este unul de lungă durată, necesitând la ora actuală câteva milisecunde.

Având în vedere multitasking-ul, MV trebuie să asigure și ***mecanisme de protecție*** a programelor (de exemplu să nu permită unui program utilizator să scrie zona de date sau cod a sistemului de operare, sau a altui program).

În implementarea MV trebuie avute în vedere următoarele aspecte:

- paginile să fie ***suficient de mari*** (4 Ko-16 Ko...64 Ko) astfel încât să compenseze timpul mare de acces la disc (9-12 ms);
- organizări care să reducă rata de PF, de unde rezultă în mod necesar un ***plasament flexibil*** al paginilor în memoria principală;
- PF-urile trebuie ***tratate prin software*** și nu prin hardware (precum *miss*-urile în chache-uri) - timpul de acces al discurilor permite lejer acest lucru;
- scrierile în MV se fac după algoritmi de tip "***write back***" (adică se scrie numai în memoria principală, nu și pe disc; scrierea pe disc se face odată cu evacuarea paginii) și nu "*write through*" (scriere în memoria principală și pe disc - procedeul ar consuma timp enorm)

Noțiuni de bază

Procese. Tabela de pagini împreună cu contorul programului (PC) și registrele microprocesorului formează împreună ***starea*** unui anumit program. Unul dintre registrele microprocesorului (numit PTR - *Page Table Register*) conține adresa la care este încărcată tabela de pagini. Se numește ***proces*** (task) un program împreună cu starea asociată acestuia. Un proces executat curent este ***activ***, altfel el este ***inactiv***.

Comutarea de task-uri constă în inactivarea procesului curent și activarea altui proces, inactiv până acum. Pentru aceasta este necesară salvarea și respectiv restaurarea stării proceselor. Desigur, sistemul de operare (SO) trebuie doar să reîncarce PTR pentru a indica tabela de pagini aferentă noului proces activ.

Excepția Page Fault (PF). Acest eveniment apare în cadrul mecanismului de translatare a adresei virtuale în adresă fizică. Dacă bitul P (pagină lipsă în memoria principală) are valoarea zero, rezultă că printr-o procedură de excepție se dă controlul unui *handler* al SO în vederea tratării. Aici, SO va trebui să decidă ce pagină din MP va trebui evacuată în vederea încărcării noii pagini de pe disc. În general aici se merge pe ideea LRU (*Last Recently Used*), adică va fi evacuată pagina care nu a mai fost accesată de către CPU de cel mai mult timp (se merge deci implicit pe principiul localității temporale).

Exemplu: CPU a accesat în ordine paginile 10, 12, 9, 7, 11, 10, iar acum vrea să acceseze pagina 8, care nu este prezentă în MP. În baza principiului LRU, rezultă că evacuează pagina 12. Dacă următorul acces generează PF, rezultă că evacuează pagina 9, ș.a.m.d.

Observație: Unele mașini (Intel 80X86, Pentium) implementează în tabela de pagini câte un ***bit de referință*** pentru fiecare pagină. Acest bit este setat la fiecare accesare a acelei pagini. SO șterge periodic acești biți - nu

Lexicon

Cache: Conceptul a fost "inventat" de către o mare personalitate în domeniul calculatoarelor, un pionier, anume profesorul Maurice Wilkes de la Universitatea din Cambridge, .
Cache este o memorie situata între procesor și memoria principală (MP), mai rapidă, mai scumpă și mai mică (în capacitate) decât aceasta. Conține, practic, o copie a datelor din MP care au fost recent utilizate de către procesor. Este gestionată hardware, astfel încât să existe o probabilitate cât mai mare ca data dorită a fi accesată de către procesor să se găsească în cache.
Memoriile cache sunt singurele care "țin pasul" cu viteza procesoarelor. În ultimii 10 ani, performanța medie a procesoarelor crește cu circa 60% pe an, în timp ce performanța (viteza) MP crește cu doar 7% pe an. Acest fapt face ca să existe un "semantic gap" între procesor și MP, care – în lipsa cache-urilor – ar face ca procesoarele să meargă cu "frâna de mână trasă".

Hit și Miss: O dată care trebuie accesată poate să se afle în memoria cache (*hit* - "succes") sau poate să nu se afle în cache (*miss* - "ratare"). Rata de *miss* în cache este procentajul din numărul total al accesele la memorie care nu au avut "succes", deci datele respective nu s-au aflat în memoria cache, ci în memoria principală. Este de dorit ca rata de *miss* să fie cât mai mică iar rata de *hit* cât mai ridicată.

Asociativitate: caracteristică a unei memorii de a permite căutarea după conținut și nu după adresă, ca în cazul memoriilor "obișnuite" (operative). Asociativitatea, în contextul ierarhizării memoriilor (cache, memorie principală, disc - memorie virtuală), permite practic oricăror locații din nivelul superior să coexiste la un moment dat în nivelul inferior de memorie. În cazul memoriilor "cache" neasociative acest lucru nu este întotdeauna posibil, existând deci locații în nivelul superior mapate pe aceeași locație din nivelul inferior (care nu pot coexista deci în nivelul inferior de memorie), ceea ce conduce la interferențe, cu efect negativ asupra performanței (rata de *miss* "mare").

înainte de a le memora starea - astfel încât să implementeze pentru fiecare pagină un contor. Pe baza conținutului acestui contor se stabilește care pagină va fi evacuată.

Scrierile. Așa cum am menționat deja, scrierea în MV se face pe baza unei strategii numite ***write-back*** (*copy-back*), deci scrierea pe disc se face doar la evacuarea paginii respective.

În practică, pentru fiecare pagină se adaugă un așa-numit ***Dirty Bit*** (D) în tabela de pagini. Bitul D este setat la fiecare scriere în pagină, astfel încât la evacuare, o pagină având bitul D=0 nu are rost să se evacueze efectiv pe disc, pentru că ar rezulta o pierdere inutilă de timp.

Translation-Lookaside Buffers (TLB). Prin paginare, fiecare acces la o anumită dată necesită 2 accese la memorie: unul pentru obținerea adresei fizice din tabela de pagini, iar celălalt pentru accesarea propriu-zisă a datei în memoria principală.

În vederea reducerii acestui timp de acces (dublu), tabela de pagini este "cașată" (memorată) în CPU. Memoria cache care memorează maparea tabelei de pagini se numește *TLB* (*Translation-Lookaside Buffer*). Ca orice cache, TLB-ul poate avea diferite grade de asociativitate. Există evacuări și încărcări între TLB și tabela de pagini din MP.

Deoarece TLB-ul este implementat în general "on-chip" (deci chiar în procesor), capacitatea sa este relativ mică (32-1024 intrări) în comparație cu tabela de pagini, care are 1-4 milioane de intrări. De obicei, TLB-ul se implementează *full-asociative*, pentru a avea o rată de *miss* scăzută (0.01% - 0.1%). *Miss*-urile în TLB se pot rezolva atât prin protocol hardware cât și printr-un *handler* software.

Protecția memoriei

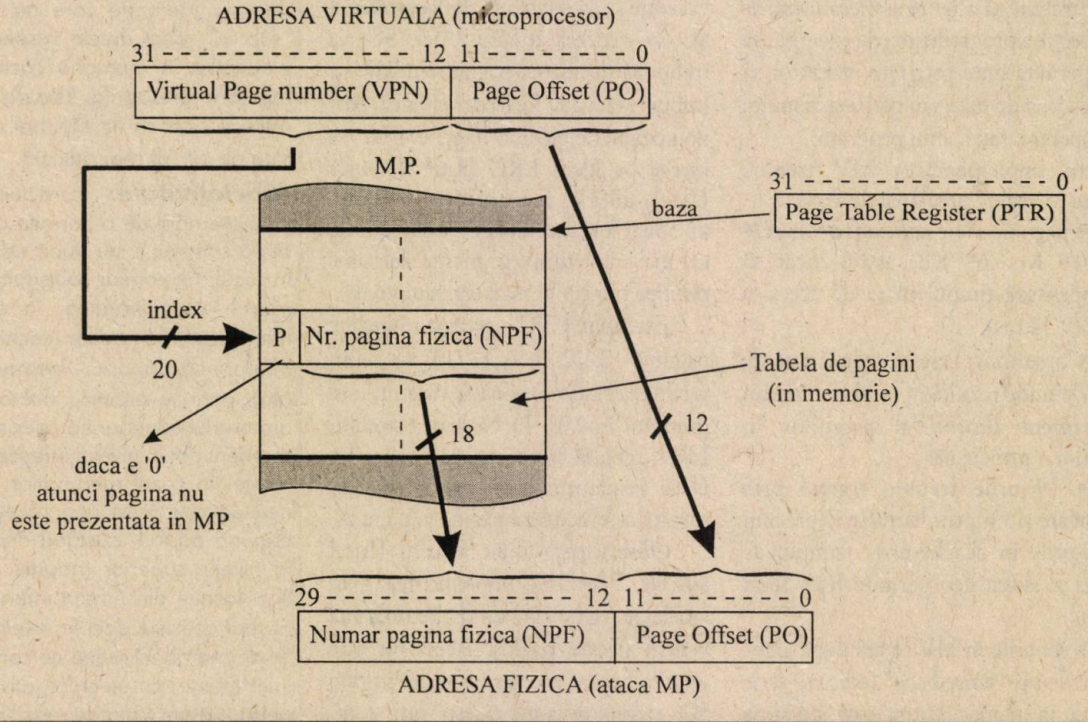
Deși fiecare proces are propriul său spațiu virtual de adresare, memoria fizică (MP) este partajată între mai multe procese (procese utilizator și procese ale sistemului de operare). Desigur, trebuie să se controleze strict accesul unui proces în zonele de cod și date ale altui proces, deci rezultă ***necesitatea protecției*** la scrieri și la citiri. De exemplu, numai SO trebuie să poată modifica tabelele de pagini aferente diferitelor procese.

În vederea implementării protecțiilor, hardul trebuie să asigure cel puțin următoarele ***3 condiții***:

- cel puțin 2 moduri distincte de rulare a unui program: ***modul supervizor*** (numit și *kernel*, sau *executiv*), în care un proces poate să execute orice instrucțiune și să acceseze oricare dintre resurse; și ***modul user*** (utilizator), în care un proces are o mulțime de restricții legate de protecția și securitatea sistemului.
- să existe o parte a stării CPU în care ***un proces user să nu poată scrie***. De exemplu: biții user/supervizor, registrul PTR, bitul de validare/invalidare a excepțiilor etc. Posibilitatea unui proces user să scrie astfel de resurse ar determina SO (proces supervizor) să nu poată controla procesele user.

Mecanisme de ***tranziție a procesorului*** din modul supervizor în modul user și invers. Tranziția user-supervizor se poate face printr-o excepție (întrerupere) sau printr-o instrucțiune specială privilegiata de tip *system call*, care transferă controlul la o adresă

Mecanism tipic de translatare



dedicată din spațiul de cod supervizor (*call gate*, la Pentium). Se salvează PC-ul și contextul procesului curent iar CPU e plasat în modul supervizor. Prin *return*, se restaurează modul de lucru anterior (în acest caz, user). De asemenea, din "supervizor" se poate trece în "user" prin simpla modificare a biților de mod (este permis!).

Exemplu: Să presupunem că un proces P2 dorește să îi transmită (citire) anumite date, printr-o pagină proprie, unui alt proces P1. Pentru aceasta, P2 ar putea apela o rutină a SO (printr-un *system call*), care la rândul ei (privilegiată!) va crea o intrare în tabela de pagini a lui P1 care să se mapeze pe pagina fizică pe care P2 dorește s-o pună la dispoziție. SO (supervizor) poate să utilizeze bitul "*Write Protection*" pentru a împiedica procesul P1 să altereze respectiva pagină. Și alți biți de tip "***drepturi de acces***" în pagină pot fi incluși în tabela de pagini și în TLB.

Observație: În cazul unei comutări de task-uri de la procesul P1 la procesul P2, TLB-ul trebuie golit din 2 motive: în caz de *hit* P2 să nu utilizeze paginile lui P1 și respectiv să se încarce în TLB intrările din tabela de pagini a procesului P2 (indicată de noul PTR). Aceasta se întâmplă numai dacă P1 și P2 folosesc VPN-uri identice (biții 31-12 din adresa virtuală). Pentru a nu goli TLB-ul prea des, se preferă adăugarea la TAG-ul acestuia a unui câmp *PID* (*Process Identifier* - identificator al procesului), care va contribui la *hit* Această informație (PID) este ținută într-un registru special, ce va fi încărcat de către SO la fiecare comutare de task-uri, rezultând ca se evită în majoritatea cazurilor golirea (și, implicit, reumplerea) TLB-ului.

În concluzie, protecția este asigurată prin:

- moduri de lucru CPU de diferite nivele de privilegiu;
- control strict al SO asupra tranzițiilor din user în kernel (prin *call gates*-uri, la o anumită adresă determinată din spațiul de cod kernel);
- protecție a paginilor prin "drepturi de acces" la pagină (read only, read/write, etc.)

Miss-uri în TLB și PF-uri

Pe durata procesului de translatare a adresei virtuale în adresă fizică pot să apară două evenimente de excepție:

- ***TLB miss***, dar pagina accesată este prezentă în memoria fizică (MP)
- ***Page Fault (PF)***, adică *TLB miss* urmat de faptul că pagina dorită ***nu*** este prezentă în tabela de pagini rezidenta în MP (bit P=0).

Un ***TLB miss*** generează de obicei o procedură hardware de aducere a numărului paginii fizice din tabela de pagini. Această operație se poate implementa prin hardware, ea durând un timp relativ scurt (20-30 perioade de tact).

Tratarea PF, în schimb, necesită un mecanism de tratare a excepției care să întrerupă procesul activ, să transfere controlul rutinei de tratare PF (SO) și apoi să redea controlul procesului întrerupt. PF va fi recunoscut pe parcursul ciclurilor de acces la memorie. Deoarece instrucțiunea care a cauzat PF ***trebuie reluată***, trebuie salvat în stivă (automat) PC-ul aferent acesteia. Pentru asta, există un registru special, EPC (*Exception PC*), întrucât PC-ul propriu-zis poate să fie mult incrementat sau chiar complet altul (*prefetch*, *branch*-uri). Mai apoi, printr-un sistem de întreruperi (vectorizate) se dă controlul rutinei de tratare din cadrul SO. Aici se determină cauza excepției prin consultarea registrului "cauză excepție" iar apoi se salvează întreaga stare (context) a procesului întrerupt (regiștrii generali, PTR, EPC, registru "cauză excepție", etc.). Dacă PF-ul a fost cauzat de un "*fetch* instrucțiune" (ciclu de aducere a instrucțiunii din memorie) atunci EPC va conține adresa virtuală care a cauzat PF; altfel (*fetch data* sau *write data*), adresa virtuală care a cauzat PF trebuie calculată din însăși formatul instrucțiunii pe care PF s-a produs (PC-ul aferent acesteia e memorat în EPC).

Odată știută adresa virtuală care a cauzat PF, rutina de tratare a SO aduce pagina de pe disc în MP, după ce mai întâi a evacuat (pe baza algoritmului LRU) o pagină din MP pe disc. Cum accesul pe disc durează mii de tacte, în mod uzual SO va activa un alt proces pe această perioadă.

Segmentarea

Segmentarea constituie o altă variantă de implementare a MV, care utilizează în locul paginilor de lungime fixă entități de lungime variabilă, numite ***segmente***. În segmentare, adresa virtuală este constituită din ***două*** cuvinte: o ***bază*** a segmentului și respectiv un ***deplasament*** (*offset*) în cadrul segmentului. Datorită lungimii variabile a segmentului (1 octet - 2^{32} octeți), trebuie făcută și o verificare a faptului că adresa virtuală rezultată (Bază+Offset) se încadrează în segment.

Desigur, segmentarea oferă posibilități puternice și sofisticate de protecție a segmentelor.

Pe de altă parte, segmentarea induce numeroase ***dezavantaje*** precum:

- Sunt necesare 2 cuvinte pentru o adresă virtuală, datorită faptului că lungimea segmentelor este variabilă. Aceasta complică sarcina compilatoarelor și a programatorilor;
- Încărcarea segmentelor variabile în memorie este mai dificilă decât la paginare;
- Fragmentarea memoriei principale (apar porțiuni nefolosite);
- În mod frecvent, traficul MP-disc devine ineficient. De exemplu, pentru segmente "mici" transferul cu discul este complet ineficient, având în vedere că accesarea discului de face la nivel de sector (de obicei 512 octeți).

Există și implementări hibride, care folosesc și segmentarea și paginarea, în scopul stabilirii unui (etern) compromis între aceste două tehnici de gestionare a memoriei.

Bibliografie:

- Hennessy J., Patterson D. - *Computer Architecture A Quantitative Approach*, Morgan Kaufman Publishers, Second Edition, 1995
- Patterson D., Hennessy J.- *Computer Organization and Design: The Hardware - Software Interface*, Morgan Kaufman Publishers, Second Edition, 1998

Domnul dr. ing. Lucian VINȚAN este conferențiar universitar la Universitatea "Lucian Blaga" din Sibiu și poate fi contactat prin e-mail la adresa : vintan@cs.sibiu.ro

SUNETUL, o necesitate

de Dragoș Marian

Unul dintre cele mai importante aspecte multimedia ale unui calculator este capacitatea de a reda diferite sunete, de la simple zgomote la vorbire și muzică. Dar ce este sunetul, cum se formează el în interiorul unui calculator și ce posibilități ne oferă pentru a-l transforma?

Sunetul este o vibrație a particulelor unui mediu, capabilă să producă o senzație auditivă. Din punctul de vedere a interfeței om-calculator, există două tipuri de sunete: cele rezultate din vorbire și alte sunete (de la simple zgomote și beep-uri la muzica simfonică).

Vorbirea digitizată constituie un domeniu de studiu important, numit Digital Speech Processing. El cuprinde trei subdomenii importante:

1. Speech Recognition&Understanding: domeniu în care s-au făcut progrese mari. Calculatorul poate înțelege unele componente individuale ale vorbirii ce au un conținut semantic. Obiectivul mai greu de atins este de a crea programe care să înțeleagă conținutul mesajului vorbit transmis de către om (aici ne lovim deja de inteligența artificială).

2. Sinteza vorbirii: calculatoare ce vorbesc oamenilor. Deocamdată s-a ajuns doar la simpla creare a unor programe care la anumite evenimente redau un sunet înregistrat în prealabil. Pentru un progres mai semnificativ este necesar un progres asemănător în domeniul inteligenței artificiale.

3. Transmiterea/recepția mesajelor vorbite: comunicarea fără sau cu un ajutor limitat din partea liniilor telefonice. Progresele în acest domeniu sunt mulțumitoa-re, având în vedere zecile de servicii de net-fonie ce există pe Internet.

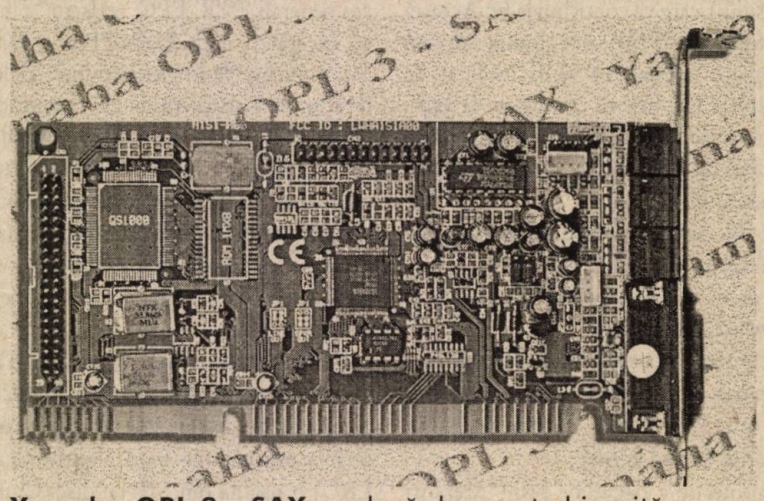
Celelalte tipuri de sunete sunt la fel de importante, însă nu beneficiază decât de clasificarea simplă în zgomote și muzică.

Sunetul este o variație pur analogă. Pentru a putea fi prelucrate cu ajutorul calculatorului este nevoie de o transpunere analogă/digitală (vezi cutia "Înregistrarea și redarea sunetului").

Hardware

În istoria calculatoarelor întâlnim zeci de variante de "interpretoare de sunet", numite plăci de sunet(Sound Card sau Sound Board). Începând cu sintetizatorul foarte limitat ca performanță inclus în toate sistemele, acel PC Speaker,

Placa de sunet



Yamaha OPL 3 - SAX: o placă de sunet obișnuită cu un raport preț/calitate destul de bun pentru utilizatorul de zi cu zi, fără pretenții profesionale.

până la plăcile de sunet pentru procesare profesională pe 128 de biți. Cele mai folosite în ziua de azi de utilizatorii normali sunt cele pe 16/32 de biți cu interfață PCI. Acestea permit înregistrarea/redarea și prelucrarea lor la standarde satisfacătoare.

O placă de sunet are mai multe componente:

1. Cipul de procesare a sunetului, diferit de la o placă la alta, este cea mai importantă componentă. De el depinde calitatea înre-gistrării/redării nonmuzicale.

2. Tabela MIDI. Sunetele pot fi înregistrate și printr-o reprezentare simbolică, cu alte cuvinte, utilizând standardul MIDI (Musical Instrument Digital Device), dezvoltat mai mult în anii 1980. Acest standard este si-milar cu cel folosit pentru a repre-zenta muzica cu ajutorul portativelor. Se definesc instrumentele (dintr-o colecție însumând 127 in-strumente diferite - piane, chitări, viori, chiar și sunete cum sunt aplauzele și ciripitul de păsărele), înălțimea, durata și secvența notelor. Deși este un standard care permite reprezentarea simbolică a unor piese instrumentale foarte lungi ocupând foarte puțin spațiu față de înregistrările digitale, calitatea redării lasă de dorit dacă este folosită o placă de sunet din care lipsește următoarea componentă.

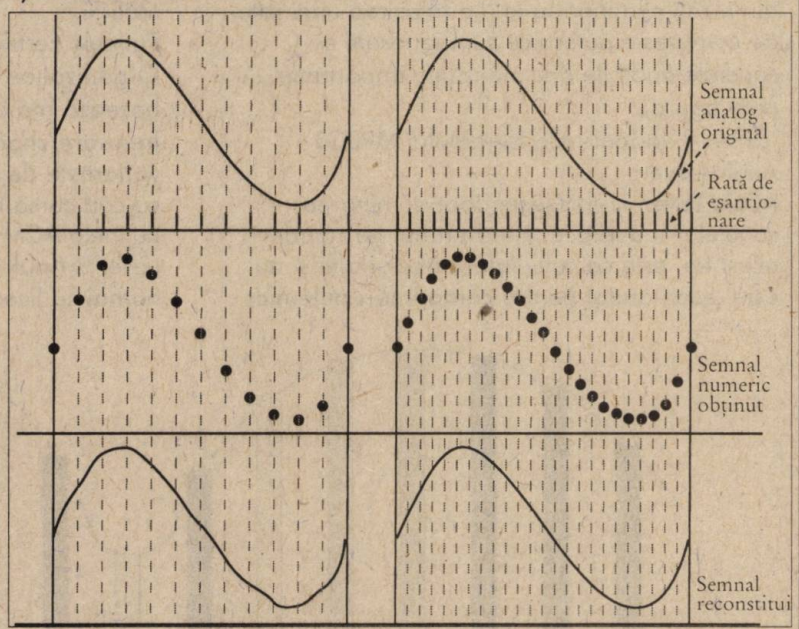
3. Tabela de instrumente (opțională). Este folosită pentru o mai bună redare a sunetelor înregi-strate cu ajutorul standardului MIDI. Este prezentă fizic sub forma unui cip pe placa de sunet sau chiar a unei extensii (o miniplacă) fixată pe placa de sunet (mai ales în cazul cumpărării ulterioare a tabelei). Conține toate instrumentele pe care le întâlnim în standardul MIDI, însă cu o prelucrare a redării mult mai apropiată de sunetul real al instrumentului, și deseori, zeci de instrumente auxiliare, nespecificate în standardul original. Compunerea muzicii pe calculator necesită în ge-neral o asemenea extensie.

Înregistrarea și redarea

Sunetul este o vibrație. Dacă ceva vibrează într-un mediu propice propagării vibrației (de obicei aerul), se creează "unde de presiune" care propagându-se, pot întâlni o structură sensibilă la schimbările mici de presiune (în cazul omului, timpanul). Astfel se formează sunetul, măsurat în două direcții: *intensitate* (decibel - dB) și *frecvență* (Hertz - notație Hz - vibrații pe secundă). Problema care se pune este cum se poate transforma această informație pur analogică în informație numerică.

Unda originală trebuie capturată (de exemplu, cu ajutorul unui microfon). Se aplică asupra ei o structură de eșantionare de o anumită frecvență. La fiecare punct de eșantionare se traduce intensitatea și înălțimea sunetului în informație digitală, stocată în memorie bineînțeles, sub formă de biți și bytes. Procesul invers este rezultat dintr-o funcție de aproximare care încearcă să reconstituie unda originală. Calitatea acestei reconstituiri depinde exclusiv de frecvența de eșantionare și de cantitatea de informație înregistrată la fiecare eșantionare. Această informație poate fi stocată pe 8 sau 16 biți. La o eșantionare de 8 biți se pot folosi 256 de niveluri pentru descrierea amplitudinii (înălțimii) unui sunet la un moment dat. Un eșantion de 16 biți oferă 65536 de niveluri, simțitor mai precis.

Neajunsul principal al digitizării este că înregistrarea nu conține o descriere semantică a sunetului. Până nu se folosesc programe de recunoaștere, calculatorul nu va ști dacă secvența digitizată este vorbire sau muzică, nu va cunoaște instrumentul și nota cântată.



Din schemă se observă cum depinde calitatea reconstituirii de rata de eșantionare.

4. Căile de intrare/ieșire a semnalului. De obicei, o placă de sunet are minimum patru canale fi-zice: *Mic* (captare semnal de la microfon), *Line In* (captare semnal de la orice aparat cu redare analogă), *Line Out/Speakers* (ieșire pentru boxe sau aparatură de amplificare, cu semnal analog) și *interfața MIDI* (prin care putem transmite o partitură MIDI unui sintetizator extern pentru redare/prelucrare). La plăcile de sunet de calitate ieșirea Line Out/Speakers este despărțită în două ieșiri distincte, ceea ce ne dă mai multe posibilități de redare/mixare. Plăcile de sunet mai au o interfață pentru jocuri (Joystick) și o interfață pentru primirea semnalului direct de la CD-ROM (CD-uri de muzică), dacă acesta este încorporat în calculator.

5. Mixerul încorporat. Este folosit pentru reglarea intensității semnalului (volum sonor) pentru fiecare din căile de intrare/ieșire descrise mai sus. Plăcile de sunet mai noi au facilități de reglare a tonalității (cantitatea de sunete joase/medii/înalte) și, uneori, a redării 3D a sunetului (bineînțeles cu boxe care să suporte această opțiune).

Sunetul și Internetul

Paginile HTML multimedia nu mai sunt de mult o raritate. Navigatoarele capabile a prelucra informații multimedia (imagini, animație, sunet) sunt acum o necesitate. Tot mai des întâlnim pagini de Web care, după încărcare, ne fac "surpriza" de a ne pune în temă cu sunete, uneori chiar cu piese muzicale.

MPEG1 Audio Layer 3

În esență, formatul MP3 este o modalitate de comprimare de până la proporția 12:1 a sunetului digitizat la frecvențe mari fără a se pierde semnificativ din calitate.

Cum comprimă?

Sunetul înregistrat digital conține secvențe inaudibile datorită acoperirii acestora de secvențe mai puternice (cu alte cuvinte, un semnal puternic, urmat de unul mai slab îl acoperă pe ultimul). Bazându-se pe un studiu al percepției auditive umane, codificatorul MPEG decide care din informații sunt elementare și care pot fi ignorate, pe acestea din urmă eliminându-le. La folosirea unei rate de compresie puternice sunt ignorate și sunetele audibile dar mai puțin importante (vezi figura).

Există 3 variante ale formatului MPEG1 Audio Layer:

Audio Layer 1: folosește doar eliminarea sunetelor inaudibile. Fiind primul format de acest tip, este calitativ inferior celorlalte și nu este recomandat pentru comprimări puternice (rate de peste 384Kbps).

Audio Layer 2: față de primul, acesta face mai multe filtrări și determină mai bine care frecvențe sunt inaudibile. Deja la 192 Kbps este dificil să simți diferența.

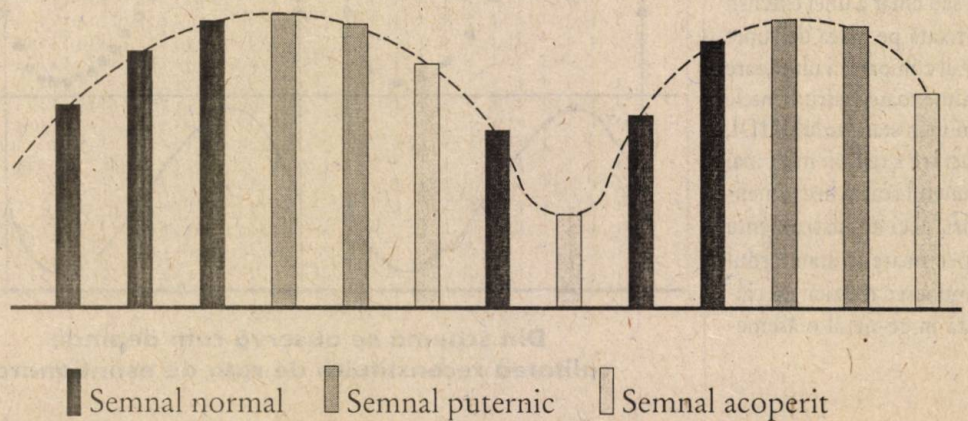
Audio Layer 3: este cel mai complex model audio MPEG. Filtrările sunt cele mai performante din această serie de formate. În plus, folosește un codificator Huffman. Deja la rate de 160-192 Kbps nu se aud diferențe față de original.

Rutine de codare MPEG

Există două rutine de bază pentru codarea MPEG:

Rutinele certificate de ISO (International Organization for Standards), pe care se bazează majoritatea programelor freeware/shareware pentru MP3. Toate variantele de codare sunt disponibile gratuit cu cod sursă la site-ul ISO.

Rutinele ACM de la FIIS, doar pentru varianta Layer 3 (mult mai bune), sunt disponibile numai cu licență.



Formatele cele mai des utilizate pentru acestea sunt Wave și MIDI. Sunetele în format Wave, mai rar întâlnite pe Internet, sunt înregistrări digitizate, de obicei de calitate mai slabă (8 biți mono) pentru a nu întârzia prea mult încărcarea paginii. Formatul MIDI despre care am vorbit deja, fiind mult mai compact, se utilizează pe o scară mult mai largă.

Dar Internetul ne oferă ceva mult mai spectaculos. Apar din ce în ce mai multe site-uri ce oferă transmisie radio în timp real în rețea. Programul RealAudio (disponibil gratuit pe site-ul www.realaudio.com) ne oferă posibilitatea de a decoda astfel de informație.

Cel mai problematic aspect al acestei decodări îl constituie inevitabilele îngustări de bandă. Cei de la RealAudio au rezolvat însă această problemă destul de elegant, cu prețul întârzierii cu câteva secunde a transmisiei reale. La pornirea programului se creează un buffer de 5-6 secunde de sunet (care reprezintă și întârzierea) ce este înnoit permanent. Sunetul auzit face parte din prima parte a acestui buffer continuu (este o structură de tip FIFO - First In First Out). În cazul întârzierii primirii semnalului prin rețea este folosit restul bufferului până când semnalul revine. Uneori, datorită unui buffer prea mic sau a întreruperii complete a semnalului (întâlnit nu foarte rar), nu putem decât să suferim la întreruperea melodiei preferate (tocmai transmisă) sau a celei mai importante știri a zilei. Un astfel de serviciu îl oferă site-ul www.netsoft.ro.

Nu mai puțin interesantă și importantă este telefonia prin Internet. Cu ajutorul unei plăci de sunet, a unui microfon și a unei conexiuni Internet putem vorbi în orice colț al globului fără a folosi rețeaua normală de telefonie. În cazul în care legătura cu Internetul se realizează cu ajutorul unui modem, singura legătură cu compania de telefoane este realizarea conexiunii la serverul de Internet pe care îl utilizați. Un astfel de program este Net2Phone. (vezi site-ul disponibil)

Pe Internet găsim sunete și muzică disponibile pentru download în cele mai

Finale 97

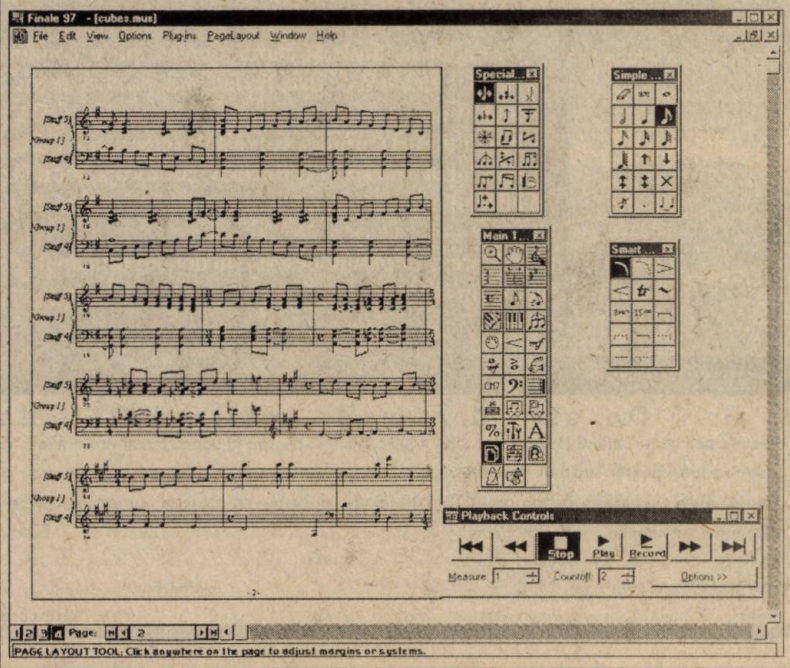
Finale 97 este un program de tehnoredactare a partiturilor deosebit de bine construit. Capabil de a crea partituri complexe, cum ar fi cele pentru orchestră, Finale 97 este unul dintre cele mai bune programe profesionale de acest tip. (demo disponibil pe site-ul www.codamusic.com)

Avantaje:

1. "Simte" mărimea grafică a notei scrise și așează automat notele următoare în funcție de aceasta.
2. Poate realiza conversia partitură MIDI/clasică și invers
3. Poate edita și tablaturi (partituri speciale pentru instrumente cu coarde cupite) și realizează rapid conversia tablatură/partitură
4. Folosește toate instrumentele extensiei MIDI
5. Folosește fonturi True-Type proprii și poate lucra cu imprimante PostScript
6. Are facilități de Copy&Paste performante
7. Pentru instrumentele inexistente în tabela MIDI pot fi definite biblioteci de caracteristici
8. Poate interpreta piesa scrisă pentru verificare.
9. Cerințele de sistem sunt mici

Dezavantaje:

1. Interfața grafică este mai greu de descifrat pentru începători
2. MeniuFrile nu sunt destul de explicite și te poți pierde repede în ele datorită multitudinii de facilități (e bine - sunt multe, e rău - nu sunt organizate bine)



diverse formate. Bineînțeles, cel mai răspândit este formatul digitizat Wave pentru Windows. Însă acesta face loc din ce în ce mai des formatului *MPEG Audio Layer 3*. Tot digitizat, dar puternic comprimat, el este o opțiune tot mai des folosită (Vezi caseta "Formatul MPEG3").

Software

Calculatorul nu mai este o raritate în domeniul creării și editării de muzică. Programele specializate în acest domeniu sunt multe, de la cele pentru copii la cele superprofesionale. Bineînțeles, toate acestea se bazează pe standardul MIDI.

Crearea muzicii pe calculator se lovește însă de posibilitățile destul de limitate ale extensiei de instrumente. Nici chiar cele mai performante plăci de sunet nu pot înlocui pe deplin un instrument clasic. Chiar dacă muzica electronică (cum ar fi compozițiile lui Jean Michel Jarre, muzica techno și rave) își ating perfecțiunea prin intermediul sintetizatoarelor și a plăcilor de sunet, nu se întrevede un viitor foarte "luminos" în domeniul reproducerii instrumentelor clasice. Și asta pentru că un interpretor MIDI nu poate înlocui interpretul clasic, muzica nefiind doar o înșiruire de note. O partitură MIDI, indiferent cât de complexă, interpretată oricât de precis de un instrument digitizat nu va egala curând interpretul uman, capabil să simtă mesajul piesei și să-și modifice interpretarea pe parcursul acesteia fără a modifica partitura originală. Calculatorul, până nu va fi dotat cu ceea ce am putea numi simț artistic (nu cred că trebuie să precizez care sunt speranțele în acest domeniu), nu va fi capabil să-și ajusteze interpretarea prin schimbări de tonalitate și ritm în funcție de piesă. Oricum, calculatoarele și-au făcut intrarea în muzică și acum puțini pot să-și imagineze o melodie stil techno sau rave cântată la instrumente clasice.

Ca o curiozitate, au apărut de curând softuri care "compun" muzică. Folosindu-se pe funcții complexe de înșiruire a notelor și de determinare a duratei acestora, bazate pe studii de teoria muzicii și de psihologie, acestea reușesc, cu ajutorul minim al utilizatorului, să creeze secvențe muzicale. Utilizatorului îi rămâne a hotărî ordinea secvențelor și viteza de interpretare a acestora, bineînțeles putând modifica orice notă.

Un domeniu important al softurilor de muzică îl constituie cel al tehnoredactării partiturilor. Un astfel de program este Finale 97, produs de Coda Music Tehnology Inc., program care se numără printre cele mai bune din domeniu.

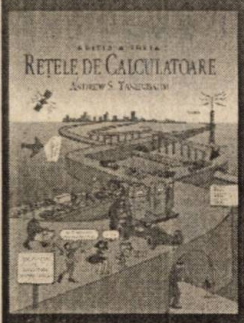
Am încercat să trasez câteva linii principale ale pătrunderii sunetului în lumea virtuală și câteva caracteristici de bază ale acestuia, însă domeniile în care sunetul digitizat își face apariția cresc de la o zi la alta. Ce ne va aduce viitorul? Principalele progrese trebuie așteptate în domeniul Digital Speech Processing, și în mărirea ratei de compactizare a sunetului digitizat.

Dragoș Marian este colaborator al revistei GInfo și poate fi contactat prin intermediul redacției: office@ginfo.ro

Prima carte românească de referință în domeniul rețelelor de calculatoare

Autor, profesor și cercetător, laureat al premiului ACM Karl V. Karlstrom Outstanding Educator Award, Andrew S. Tanenbaum, explică cu grijă felul în care rețelele funcționează în interior, pornind de la tehnologia hardware până la cele mai utilizate aplicații de rețea. Cartea urmează o abordare structurată a rețelelor de calculatoare, începând de jos (de la nivelul fizic) și parcurgând treptat calea spre vârful acestora (nivelul aplicațiilor).

Subiectele abordate includ:



- **Nivelul fizic (firele de cupru, fibrele optice, comunicațiile radio și cele prin satelit)**
- **Nivelul legătură de date (principiile protocoalelor, HDLC, SLIP și PPP)**
- **Subnivelul de acces la mediu (LAN-uri IEEE 802, punți, noile LAN-uri de mare viteză)**
- **Nivelul rețea (rutarea, controlul congestiei, interconectarea rețelelor, IPv6)**
- **Nivelul de transport (principiile protocoalelor de transport, TCP, performanța rețelei)**
- **Nivelul aplicație (criptografia, poștă electronică, știri, Web-ul, Java, multimedia)**

În fiecare capitol sunt descrise în detaliu principiile de bază, urmate apoi de exemple ample luate din Internet, rețele ATM și rețele de comunicații fără fir.

Doresc să cumpăr cartea Rețele de calculatoare

V-am expediat în contul C.P. Agora nr.: 2021400027100178 deschis la Banca Transilvania, filiala Mureș, 100.000 lei (în care nu sunt incluse taxele poștale), cu mandatul poștal/ordinul de plată nr.: ..... din data de ..... ☐ Doresc să-mi trimiteți și factură.

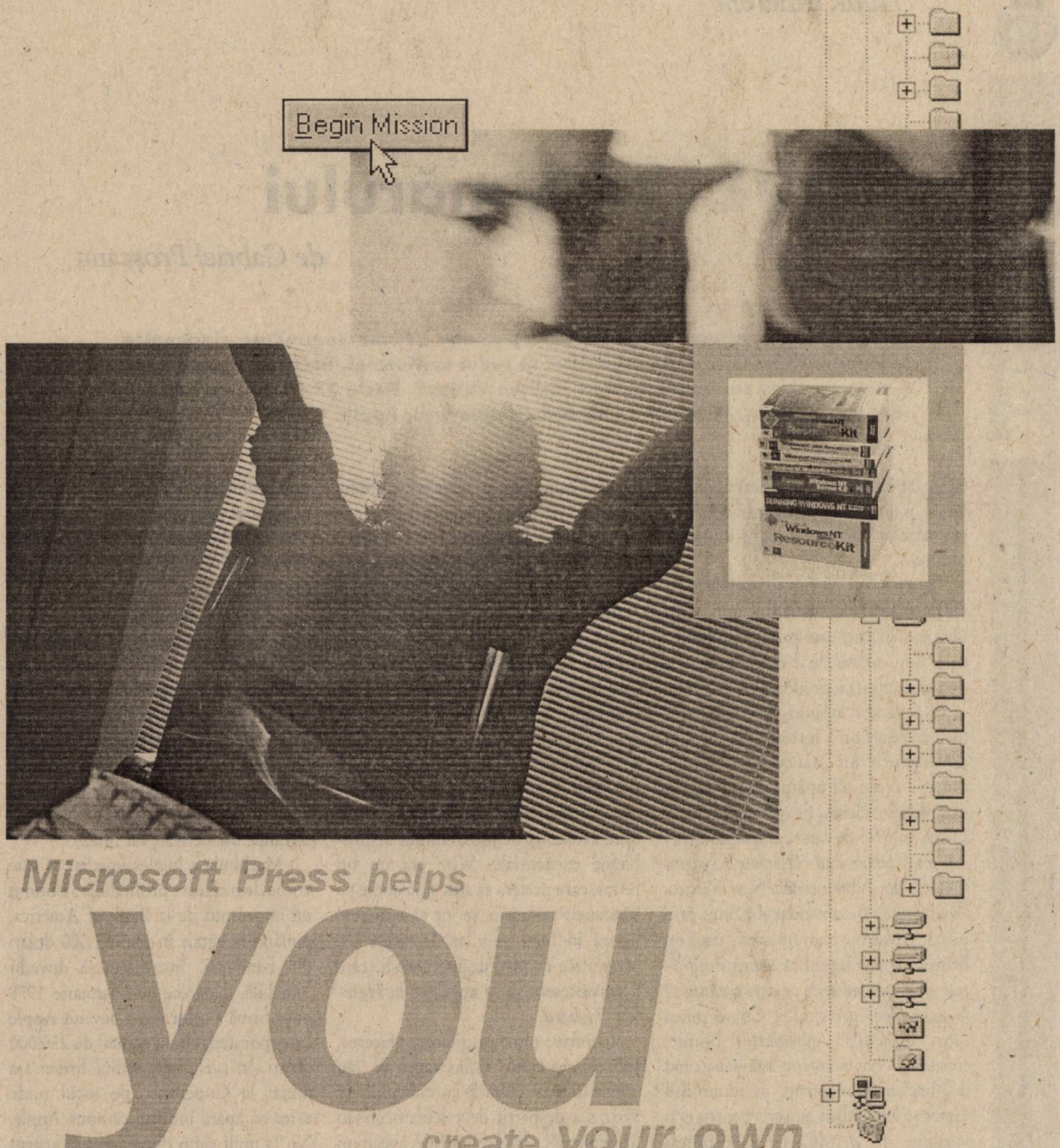
Pentru elevi și studenți cartea costă 75.000 lei. Ei vor trebui să atașeze comenzii o copie după carnetul de elev sau student.

Numele și prenumele: ..... Funcția: .....

Firma: ..... Adresa: .....

Cod poștal/Localitate: ..... Telefon/Fax: .....

Data ..... Semnătura .....



Microsoft Press helps

you

create your own answers

where do you want to go today? Microsoft Press

COMPUTER PRESS AGORA

COMPUTER PRESS AGORA

Sucursala București, Str. Poterași 35/2, Tel.01-3309281, Fax: 01-3309285

Think different

POVESTEA mărului

de Gabriel Proșcanu

În 1939, Dave Packard și Bill Packard porneau o firmă de produse electronice într-un garaj închiriat din Palo Alto, în sudul Californiei. Acest garaj este temelia a ceea ce urma să devină curând "Silicon Valley". Peste 37 de ani, alți doi tineri, într-un alt garaj, porneau o altă firmă legendară: Apple.

Se poate începe într-un garaj o afacere de 11 miliarde de dolari? În 1998 nu știu, dar în 1976 acest lucru a fost cu putință...

În acei ani '70 (nu foarte îndepărtați) o mulțime de lucruri stranii se întâmplau, lucruri care provoacă uimirea celui familiarizat doar cu computerele de astăzi, nu și cu predecesorii acestora.

Adevăratul strămoș al computerelor personale nu a fost acel PC produs de IBM în 1981 - așa cum se crede de obicei - ci un alt aparat numit **Altair**, care fusese lansat pe piață, cu mare succes, în 1975 de către compania *Micro instrumentation and Telemetry Systems* (MITS), din Albuquerque, New Mexico. Altair putea fi comandat și primit prin poștă; probleme mari nu avea - dacă nu luăm în seamă faptul că *venea sub forma unor componente pe care trebuia să le asamblezi personal*... Ce se putea face cu această mașinărie? Nimic, trebuie s-o recunoaștem; asta până când doi hackeri adolescenți, pe nume **Bill Gates** și **Paul Allen** au scris pentru el o versiune de BASIC, cu care puteai realiza calcule rudimentare...

Tot în 1975, un tânăr numit **Steve Jobs** se chinuia să-și afle un țel în viață, acum că terminase liceul; fusese în India și se întorsese; fusese la o fermă în Oregon și se întorsese; acum își pierdea timpul la Atari, ca programator pentru jocuri video și frecventa un club de pasionați ai computerelor, care își ținea reuniunile într-un parc, pe timpul nopții...

Când Altair apăruse, membrii clubului au fost fericiți că făcuseră rost de o nouă jucărie; Jobs, însă, a văzut aici o modalitate de a câștiga ceva bani. Imediat s-a gândit să facă și el un asemenea computer. Excelentă idee, numai că habar nu avea cum...De aceea, a apelat la un amic - **Steve Wozniak** ("Woz") - care tocmai scrisese un limbaj de programare (să nu mă întrebați cum se numea acest limbaj... probabil că nimeni nu știe acest lucru, în afara celor doi Steve) pentru un microprocesor nou, numit MOS Technology 6502. Spre deosebire de Jobs, chinuit de probleme existențiale, Woz era un tip retras, care prefera să stea și să apese pe butoanele tastaturii, în loc să cutreiere lumea în lung și-n lat. În acest fel, acumulase cunoștințe inginerești, care îi permiseseră să se angajeze la *Hewlett-Packard*.

Pentru a pune pe picioare afacerea, Jobs și-a vândut Volkswagen-ul său vopsit în toate culorile curcubeului, iar Woz s-a despărțit de calculatorul său programabil. Părinții lui Jobs aveau un garaj în Los Altos, California; acolo și-au stabilit cei doi cartierul general - garajul le servea drept cameră a proiectanților, unitate de service și magazin, în același timp... După câtva timp, au realizat 50 de plăci de bază, proiectate de Woz, pe care le-au vândut amatorilor locali, la prețul de 500 de dolari bucata. Noile jucării au fost botezate **Apple**, după numele casei de discuri care producea melodiile Beatles, grupul preferat al lui Jobs... Întâi aprilie este ziua păcălelilor, dar *1 aprilie 1976* a fost și o zi istorică, ziua în care s-a vândut primul Apple...

Vânzările mergeau bine, atât de bine, încât cei doi și-au dat seama că nu mai erau competenți să conducă o afacere care lua proporții; de aceea, l-au cooptat pe proaspătul pensionar **Mike Markkula**, care reușise invidiata performanță de a ieși la pensie la vârsta de 32 de ani! (Acest lucru fusese posibil după câteva manevre pline de succes cu acțiunile bursiere ale lui Intel.)

Markkula a înțeles imediat că este vorba de o afacere profitabilă; a obținut un împrumut de la Bank of America, punând la bătaie în plus 91.000 dolari din banii săi. Investiția s-a dovedit rentabilă, deoarece pe 3 ianuarie 1978 mica firmă a celor trei a devenit **Apple Incorporated**, cu un capital de 250,000 dolari. În acele zile, sediul firmei s-a mutat la Cupertino, pe locul unde urma să apară întinsul campus Apple. Nu la mult timp după aceea, a apărut Apple II, care era ceva mai mult decât o simplă placă de bază - avea cutia sa proprie, sursă de alimentare, o tastatură încorporată... Multe elemente noi și atractive apăruseră, ca de pildă posibilitatea de a conecta computerul la un monitor color. Nu e de mirare deci, că la *West Coast Computer Fair*, în aprilie 1978, Apple II a devenit cea mai populară mașină, iar vânzările au atins

cifra de 300 de milioane de dolari...

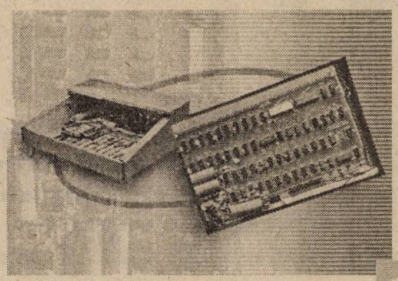
Succesul acestui tip de aparate a atras în competiție nenumărate firme. *Atari, Zenith, Commodore, Tandy* - au început să producă tot felul de computere similare și *incompatibile*. Adevărata problemă a apărut însă atunci când în ring a coborât IBM. Nu cu mult entuziasm, e adevărat - IBM vindea *mainframes* cu cel puțin un milion de dolari bucata și avea o cifră de afaceri de 26 miliarde anual, așa că vânzarea unor mizilicuri de nici 5.000 de dolari nu părea foarte atractivă. Cu toate acestea, piața computerelor personale era prea dinamică pentru a putea fi neglijată, așa că IBM a scos în 1981 un aparat destul de rahitic, numit *PC*... Pentru Apple, aflat în plin avânt, acest lucru nu a avut vreo semnificație deosebită; chiar dacă era IBM, în acel moment nu era decât un competitor printre mulți alții, aflați cu mult în urma Apple. Și asta deoarece Apple scosese un model revoluționar: *Macintosh*.

Diferențele între lumea PC-urilor IBM și a celor produse de Apple se adâncise; deși primele se numeau "computere personale", cu greu puteai găsi aparate mai impersonale și mai neprietenoase. Până și forma lor colțuroasă contrasta cu rotunjimile Macintosh-urilor. Oricine pornea computerul dădea de un ecran negru; trebuia să scrii comenzi criptice și era suficient să greșești o literă, pentru a fi lămurit cu replici pline de înțeles ca "Invalid command"... Macintosh-urile erau cu totul altfel; când porneai aparatul, dădeai de mutra veselă a unui Mac; dacă încercai să faci vreo bazaconie, măcăia ca o rață...

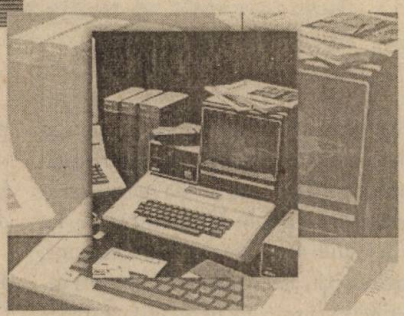
"Arată bine. Îl simt că este Mac-ul meu", spunea **Cheryl England**, redactor al unei reviste numite *Mac Addict*. "Mă binedispune și mă face doritor să lucrez cu el..."

Hmmm... nu prea auzeai astfel de remarci de la utilizatorii de PC-uri... Diferența fundamentală dintre cele două aparate era prezența unei interfețe grafice (GUI) la cele produse de Apple. Despre aceste interfețe, rolul lor și cum anume au revoluționat ele lumea computerelor veți afla într-un număr viitor.

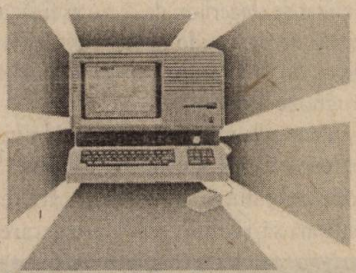
1974: Steve & Steve
Totul a început în 1974 la Homebrew Computer Club, în California. După ce călătorise în India în căutarea iluminării spirituale, Steve Jobs se întoarce în Silicon Valley și îl reîntâlnește pe Steve Wozniak.



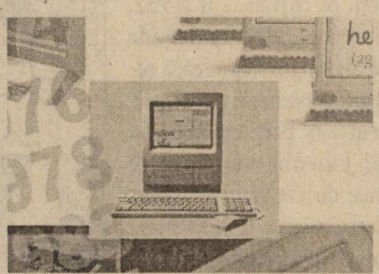
1976: Apple I
Prototipul primului calculator Apple a fost construit în garajul lui Steve Jobs. Pentru a fabrica prima serie, Jobs și-a vândut microbuzul Volkswagen iar Woz calculatorul științific Hewlett-Packard. Apple a pornit cu 1.300 de dolari.



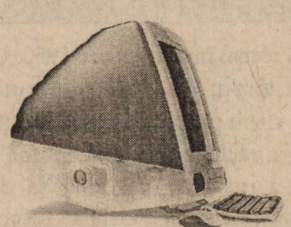
1978: Apple II
Reproiectat din temelie, Apple II stabilește un standard în domeniul microcomputerelor și aduce companiei venituri de 139 de milioane de dolari.



1983: Lisa
În 1983, Lisa oferă pentru prima dată o interfață grafică și un mouse, deschizând o nouă eră în industria informatică. Ceea ce nu împiedică însă eșecul comercial al produsului.



1984: Macintosh
În 1984 apare primul Macintosh, computerul zâmbitor și prietenos. Apariția Mac-ului în lumea calculatoarelor a fost echivalentul apariției telefonului în lumea telegrafului. Revoluția a început.



1998: iMac
"Cel mai original Macintosh de la Macintosh-ul original încoace". Dotat cu un procesor RISC PowerPC G3 la 233 MHz, un disc de 4 GB, modem de 56 K și tot ce și-ar putea dori un utilizator pretențios...

Primul calculator comercial

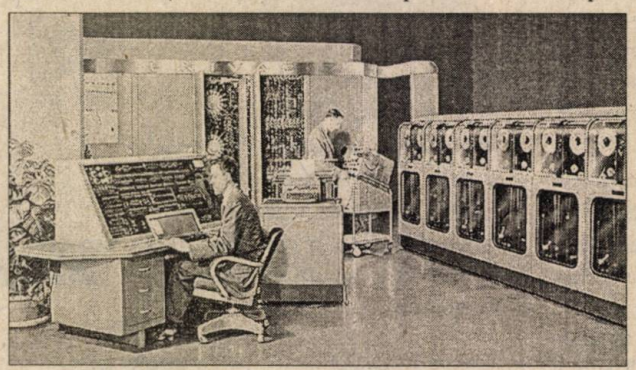
UNIVAC I

de Gabriel Proşcanu

Poate vi se pare straniu să vorbim despre "istoria" unui domeniu tehnologic atât de tânăr cum este informatica. Cei mai mulţi dintre pionierii calculatoarelor sunt contemporanii noştri, dar minunatele lor maşini au luat demult drumul muzeelor... Însă tehnologia seamănă cu moda - ideile fundamentale se reîntorc din când în când - aşa că, poate, câteva repere pot fi folositoare.

Deşi primele computere apăruseră în laboratoarele americane pe la mijlocul anilor '40, câţiva ani nu s-a reuşit transformarea lor în produse comerciale. La ce ar fi folosit aceste maşinării enorme, care ocupau câteva camere şi pentru întreţinerea cărora era necesară o întreagă echipă?

În 1952 o asemenea ciudăţenie, realizată de Eckert şi Mauchly, purtând straniul nume de Univac I, începea să fie oferită spre cumpărare potenţialilor clienţi de către compania Remington Rand. Asemenea clienţi nu prea existau, iar aparatul făcea tot felul de demonstraţii insipide, în faţa puţinelor persoane interesate de aparate de acest gen. Şi totuşi... în 1952, Univac I, folosind metode de eşalonare statistică, a realizat o estimare electorală, prin care prezicea victoria lui Eisenhower, înainte chiar de a se închide urnele. Estimarea a fost trimisă imediat reţelei de televiziune CBS; şeful departamentului de ştiri a refuzat categoric să transmită pe post "o asemenea aiureală"... Şi totuşi, rezultatele oficiale au confirmat previziunile Univac!



Lecţia poate fi utilă şi pentru programatorii de astăzi: puterea de calcul a minunii de Univac era ridicol de scăzută, faţă de aparatele de care dispunem astăzi; algoritmii folosiţi, modul de programare şi eficacitatea codului sînt factori mult mai importanţi decât puterea de calcul...

Primii cumpărători

Ca urmare a acestei întâmplări, interesul oamenilor de afaceri a crescut imens. Dacă aparatul fusese în stare să prevadă rezultatele alegerilor, de ce nu ar fi făcut acelaşi lucru şi în domeniul vânzărilor? De ce nu ar fi putut realiza şi analiza prognoze economice?

Primul cumpărător al unui Univac I a fost, în 1953, Oficiul de Evidenţă a Populaţiei din Statele Unite, care intenţiona să-l folosească la prelucrarea datelor recensământului din 1950. Un an mai târziu, în 1954, a apărut şi primul client privat - General Electric Co., care a achiziţionat un Univac I pe care l-a instalat în uzina sa din Louisville, Kentucky. General Electric producea tot felul de bunuri de larg consum - maşini de spălat, frigidere, cuptoare, tot felul de aparate menajere... Datorită boomului economic din acei ani, General Electric s-a decis să automatizeze procesul de producţie; au folosit Univac nu numai la procesarea statelor de salarii şi evidenţelor contabile, ci şi la planificarea producţiei şi la controlul fluxului tehnologic.

Aplicaţii

Angajaţii serviciului de contabilitate de la General Electric au fost cei care au avut sarcina de a crea programul de plăţi, bazându-se pe nucleul dezvoltat de cei de la departamentul maşinilor de spălat şi uscătoarelor...

Iniţiativa respectivă nu s-a dovedit a fi floare la ureche, aşa că au fost chemaţi şi oameni de la Univac. Printre aceştia se afla şi Burton Grad, care descrie astfel toată povestea:

"Mi-am petrecut trei luni la Louisville, proiectând şi programând aplicaţiile respective - facturarea materialelor, planificarea procesului de producţie, analiza inventarului şi a ca-pacităţii de producţie. Aceasta se întâmpla în iarna şi primăvara lui 1995. După ce am terminat, m-am reîntors la New York şi mi-am petrecut vara curăţând de scame programele pe care tocmai le scrisesem..."

Exces de zel? Nici pomeneală. Pentru a înțelege de ce nu exista altă cale mai rapidă, să-l ascultăm pe Burton descriind computerul cu care avea de-a face:

"În 1954, Univac era compus din:

- O unitate de calcul;
- Memorie internă și intermediară (tuburi cu mercur);
- Suport extern de stocare a datelor (benzi magnetice din metal);
- Imprimante;
- Cititoare de cartele;
- Aparatul avea următoarele facilități de programare: Nici una [!!!]"

Programare

Ce vremuri... Programele erau scrise direct în cod mașină, cu ajutorul unui cod de instrucțiuni (A însemnând "add", S "subtract" etc.) și a unor numere care nu erau altceva decât adresa fizică din memorie unde avea să fie executată operația! Aplicațiile rezultate nu erau tocmai mititele... datorită faptului că o instrucțiune putea lucra cu o singură adresă de memorie, simpla operație de înmulțire a două numere necesita trei instrucțiuni (se încărca un registru intern cu un număr, apoi se înmulțea numărul din acel registru cu alt număr și se încărca alt registru cu rezultatul, apoi conținutul acestuia din urmă era scris într-o locație de memorie, de unde putea, în sfârșit, să fie folosit...)

Exasperant? Dar aceasta nu era decât partea plăcută a programării unei asemenea mașini, dacă ne gândim și la gestionarea intrărilor și ieșirilor (I/O)... E adevărat că nu existau decât 2000 de locații - în tuburile cu mercur - în care se stocau informațiile respective, dar acestea erau adrese absolute! Astfel încât orice schimbare a unor înregistrări sau a unor câmpuri, însemna schimbarea unei asemenea adrese, adică reprogramarea întregii aplicații... În plus, pentru o eficiență sporită, trebuia să se țină cont și de faptul că o adresă era disponibilă doar de cinci ori pe secundă, iar acest lucru cerea un efort sporit de sincronizare a codului cu accesul I/O...

Lecția învățată

Depanarea programelor și serviciul tehnic pentru asemenea mașini se făcea în afara orelor de producție, adică între 6 după-amiaza și 12 noaptea. Evident că nu era o plăcere deosebită să faci asemenea operații, dar nu-i așa că veți fi surprinși dacă vă voi spune că Univac avea și difuzoare, iar programatorul de serviciu studia aplicațiile ascultând muzică? Cu toate acestea, nu multimedia a fost beneficiul lucrului cu Univac, ci altceva, mult mai important, așa cum concluzionează și Burton Grad:

"Ca urmare a acestor proiecte, neam dat seama că nimeni nu e capabil să prevadă cât timp va fi necesar sau cât efort va trebui depus pentru proiectarea, dezvoltarea și testarea unor aplicații. Am început să înțelegem că fără limbaje de nivel înalt (adrese relative, instrucțiuni compuse, operatori *if-then*...), fără depanatoare (*debuggers*) și instrumente de validare, nu vom putea produce ușor programe de calitate, iar aplicațiile rezultate nu vor putea fi ușor de întreținut, depanat sau dezvoltat..."

Într-adevăr, epoca eroică a computerelor era pe cale de a se încheia. Toate dezideratele lui Burton urmau să se împlinească rapid, dar despre acestea, într-un articol viitor.

Gabriel Proșcanu este redactor la BYTE România și PC Report. Poate fi contactat prin e-mail la adresa gproscanu@agora.ro.

COMPUTER PRESS AGORA

Prin intermediul firmei noastre puteți primi în mod regulat oricare dintre revistele americane de calculatoare din tabelul alăturat. Pentru aceasta este suficient să ne trimiteți o comandă în care să specificați titlurile dorite și datele Dvs. de identificare (nume, adresă, telefon, fax), iar noi vă vom expedia cu plata prin ramburs revistele solicitate. Prețul în lei pe care îl veți achita la primirea fiecărui colet poate fi obținut prin înmulțirea prețului în dolari specificat în tabel cu cursul de schimb valutar al BNR la care se adaugă cheltuiala de expediție percepută de Poșta Română. Atenție deci,

NU TRIMITEȚI BANI ÎN AVANS!!!

Vă rugăm să expediați comenzile Dvs. pe adresa:

Computer Press Agora srl, CP 94 OP 49, București, fax: 3309285

TITLU	FRECVENȚĂ	PREȚ
3D DESIGN	12	5.93
ACCESS OFICE VBA ADVISOR	12	7.49
AIXTRA	4	7.43
BOARDWATCH	12	8.93
BYTE	12	5.93
C/C++ USERS JOURNAL	12	7.43
C++ REPORT	9	11.93
UNIÇENTER TNC ADVISOR	6	8.93
CADALYST	12	5.93
CADENCE	12	5.93
CD-ROM PROFESSIONAL	12	10.43
CIRCUIT CELLAR	12	5.93
COMPUTER ARTIST	6	7.43
COMPUTER GAMING WORLD	12	5.99
COMPUTER GAMING WORLD wCD	12	11.99
COMPUTER GRAPHICS WORLD	12	7.43
COMPUTER PLAYER	12	10.43
COMPUTER SHOPPER	12	7.49
COMPUTER SHOPPER BUYERS CD		7.49
CYBER SPORTS	4	7.49
DATA BASE-WEB ADVISOR	12	7.49
DATABASE PROG. & DESIGN	12	7.43
DBMS	12	5.93
DBMS 1996 BUYERS GUIDE		7.43
DELPHI INFORMANT	12	8.93
DIGITAL VIDEO MAGAZINE	12	7.43
DR. DOBBS JOURNAL	12	7.43
ELECTRONIC PUBLISHING	12	7.43
ELECTRONIC GAMING MONTH	12	7.49
EMG2	12	7.49
FILE MAKER ADVISOR	24	7.99
FOXPRO ADVISOR	12	7.49
GAMEFAN	12	8.93
GAME DEVELOPER	6	8.93
ULTRA GAME PLAYER	12	7.49
GAME PRO	12	7.49
INTERACTIVITY	12	7.43
INTERNET WORLD	12	7.43
INTERNET JAVA & ACTIVEX ADVISOR	6	10.49

TITLU	FRECVENȚĂ	PREȚ
JAVA DEVELOPERS JOURNAL	12	11.93
JAVA PRO	4	10.43
JAVA REPORT	6	7.43
LAN MAGAZINE	12	7.43
LINUX JOURNAL	12	6.00
LOTUS NOTES ADVISOR	12	10.49
MACADDICT	12	11.99
MACTECH	12	8.98
MACWORLD	12	8.98
MAGAFAN	6	5.93
MICROSOFT INTERACTIVE DEV	12	7.43
MICROSOFT SYSTEMS JOUR.	12	7.43
MONDO 2000	4	8.93
NEW MEDIA	12	7.43
OBJECT MAGAZINE	6	6.75
OBJECT ORIENTED PROGRAM.	9	13.50
PC COMPUTING	12	5.99
PC GAMER	12	5.99
PC GRAPHICS & VIDEO	12	7.43
PC MAGAZINE	22	5.99
PC MAGAZINE CD-ROM	4	26.93
PC MAGAZINE SPECIAL ISSUE	4	8.93
PC WORLD	12	8.93
PRACTICAL WINDOWS	6	8.93
PUBLISH	12	7.43
SOFTWARE DEVELOPMENT	12	5.93
SYS ADMIN	6	7.43
UNIX REVIEW	12	5.93
VB TECH JOURNAL	12	5.25
VIDEO TOASTER USER	6	7.43
VISUAL BASIC PROG. JOUR.	12	5.93
VISUAL DEVELOPER MAGAZINE	6	7.43
WEB, THE	12	5.93
WEB DEVELOPER	6	8.93
WEB TECHNIQUES	12	7.43
WINDOWS/DEV DEVELOPERS	12	7.43
WINDOWS NT	12	7.43
WINDOWS SOURCES	12	4.49
WINDOWS TECH JOURNAL	12	6.00
WIRED	12	7.43

Revistele pot fi răsfoite, comandate și cumpărate de la librăria noastră deschisă la sediul firmei ***Forte Computers:*** *str. Lipscani nr. 102; precum și de la următoarele magazine din* ***București:*** *Flamingo Computers, Bd.N. Titulescu nr.102; MIVAS: Pasaj Universitate; Raffles Computers Shop, Calea Victoriei nr.25; Librăria "Noi", sala Dales; Librăria Humanitas din Pasajul Cretzulescu (lîngă magazinul Muzica);* ***BRAȘOV:*** *Interprof International, str.Operetei nr.86;* ***TIMIȘOARA:*** *COMPUTER 21 SRL B-dul Regele Ferdinand nr. 2;* ***ARAD:*** *BB Computer, B-dul Revoluției 26-38.*

Puteți comanda orice carte de la: ***Computer Press Agora s.r.l.****, C.P. 94 OP.49, București, fax: 01-3309285*

Important: Lista completă de prețuri poate fi obținută via BBS 065-210780 (login: bbs, terminal: ANSI, ...)

ALGORITMI geometrici elementari

Grigore ALBEANU

Pornind de la câteva rezultate simple de geometrie analitică plană, sunt prezentate metode algoritmice pentru rezolvarea problemelor privind intersecția segmentelor de dreaptă și a poligoanelor. În final este abordată problema punctului interior. Pentru toate temele se discută și aspecte relative la complexitatea rezolvării.

1. Preliminarii

1.1. Elemente de geometrie analitică

Fie un *plan P* și *(Ox, Oy)* o pereche ordonată de *axe ortogonale* în planul *P*,

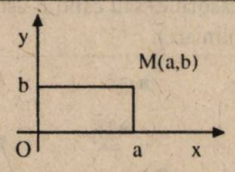


Fig. 1: *Sistem de axe ortogonale în planul P*

cu *originea O* comună și cu sensurile pozitive ale axelor indicate prin săgeți ca în fig. 1. Acest ansamblu este numit *reper cartezian* în planul *P*, este notat cu *xOy*, iar *O* este originea reperului. Pentru orice punct *M* din planul *P* este definită funcția bijectivă $f : P \to R_2$, $f(M)=(a,b)$ care asociază punctului *M* coordonatele carteziene *a* și *b*.

Următoarele rezultate vor fi considerate elementare:

a) *Formula distanței*: Fie $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$ puncte ale planului *P* specificate în reperul cartezian *xOy*. Dacă prin M_1M_2 notăm distanța dintre punctele M_1 și M_2, atunci

$$M_1M_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

b) *Formula pantei unei drepte*: Fie *d* o dreaptă neparalelă cu axa *Oy* pe care fixăm două puncte distincte $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$. Panta (coeficientul unghiular al) dreptei *d* este dată de formula

$$m = \frac{y_2 - y_1}{x_2 - x_1}, x_1 \neq x_2.$$

c) *Caracterizarea dreptelor oblice paralele/perpendiculare*: Dreptele oblice d_1 și d_2, de pante m_1 respectiv m_2 sunt paralele dacă și numai dacă $m_1=m_2$. Ele sunt perpendiculare dacă și numai dacă $m_1m_2=-1$.

d) *Ecuația dreptei oblice*: Fie dreapta *d* definită prin punctul $M_0(x_0,y_0)$ situat pe *d* și prin panta *m*. Atunci punctul $M(x,y)$ aparține dreptei *d* dacă și numai dacă $y-y_0=m(x-x_0)$. Fie dreapta *d* definită prin punctele $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$. Dacă $x_1=x_2$, atunci dreapta *d* este verticală și are ecuația $x=x_1$. Dacă $y_1=y_2$, atunci dreapta *d* este orizontală și are ecuația $y=y_1$. Dacă $x_1 \neq x_2$ atunci dreapta *d* este oblică, iar ecuația acesteia poate fi scrisă în oricare din formele echivalente:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

sau

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

sau (exprimarea robustă)

$$(x - x_1)(y_2 - y_1) - (y - y_1)(x_2 - x_1) = 0.$$

e) *Condiția de coliniaritate a trei puncte* $M_i(x_i,y_i)$, $i = 1, 2, 3$:

$$(x_3 - x_1)(y_2 - y_1) - (y_3 - y_1)(x_2 - x_1) = 0.$$

f) *Rapoarte*: Fie $M_1(x_1,y_1)$ și $M_2(x_2,y_2)$ două puncte diferite care determină segmentul M_1M_2 si dreapta (M_1M_2). Fie *M* pe dreapta (M_1M_2) diferit de M_2, $M(x,y)$. Numărul *k* definit prin regula:

Dacă M aparține segmentului M_1M_2 *atunci:*

$$k = -\frac{MM_1}{MM_2}$$

$$k = \frac{MM_1}{MM_2}$$

altfel

se numește *raportul în care punctul M împarte segmentul nenul M_1M_2 ($k\neq1$)*. Fiind cunoscut numărul real $k\neq1$, coordonatele (x,y) ale punctului M care împarte segmentul M_1M_2 în raportul k sunt:

$$x = \frac{x_1 - kx_2}{1-k},\ y = \frac{y_1 - ky_2}{1-k}.$$ s

g) *Ecuațiile parametrice ale dreptei*: Fie M_1, M_2 ca mai sus. Egalitățile:

$$x = x_1 + (x_2 - x_1)t;\ y = y_1 + (y_2 - y_1)t, t \in \mathbb{R}$$

se numesc ecuațiile parametrice ale dreptei (M_1M_2).

Dacă impunem $t\in[0,1]$ atunci obținem segmentul M_1M_2. Mijlocul segmentului M_1M_2 se obține pentru $t=1/2$.

h) *Centrul de greutate*: Centrul de greutate al unui sistem omogen de n puncte: M_1, M_2, ..., M_n cu $M_i(x_i,y_i)$, $i=1, 2, ..., n$, are drept coordonate: media aritmetică a absciselor, respectiv media aritmetică a ordonatelor.

i) *Distanța de la un punct la o dreaptă:* Formula distanței de la punctul $M_0(x_0,y_0)$ la dreapta $h: ax+by+c=0$ este:

$$d(M_0,h) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2+b^2}}.$$

j) *Aria unui triunghi*: Fie $M_i(x_i,y_i)$, $i=1,2,3$, puncte necoliniare. Aria triunghiului $M_1M_2M_3$ este dată prin:

$$\sigma(M_1M_2M_3) = \pm\frac{1}{2}[(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)].$$

Semnul se ia + sau - astfel încât aria triunghiului să fie pozitivă. Dacă M_1, M_2, M_3 sunt specificate în ordinea inversă a acelor de ceasornic atunci semnul este +. Deci, condiția ca trei puncte M_1, M_2, M_3, date prin coordonatele carteziene (x_i,y_i), să fie specificate în sens invers acelor de ceasornic se scrie:

$$(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) > 0.$$

k) *Semiplane*: O dreaptă d din planul P are proprietate că separă planul P în două submulțimi. Fie $f:R_2\to R$, $f(x,y)=ax+by+c$. Definim $P-$ – mulțimea punctelor pentru care $f(x,y)<0$, iar $P+$ – mulțimea punctelor pentru care $f(x,y)>0$. Evident, d este mulțimea punctelor pentru care $f(x,y)=0$. Mulțimile: $P-$, $P+$ se numesc *semiplane deschise*. Dreapta d se numește *frontiera semiplanelor*. Deoarece semiplanele sunt mulțimi convexe, orice intersecție de semiplane este o mulțime convexă. Pentru aflarea semnului unui semiplan se utilizează metoda sondajului: se alege un punct particular (x_0,y_0) și se vede semnul numărului $f(x_0,y_0)$. Semiplanul opus are semn contrar.

1.2. Câteva funcții utile

În această secțiune sunt prezentate două funcții elementare, utile în secțiunile următoare.

a) Funcția s

Fie d o dreaptă determinată de două puncte $A(x_A,y_A)$ și $B(x_B,y_B)$. Fie Q un punct oarecare din planul celor două puncte. Notăm prin $s(Q,A,B)$ funcția semn a punctului Q relativ la dreapta (AB), $s(Q,A,B)=1$ dacă $Q\in P^+$, $s(Q,A,B)=-1$ dacă $Q\in P^-$, respectiv $s(Q,A,B)=0$ dacă $Q\in(AB)$.

Conform celor prezentate mai sus, notând

$$E := (x_Q - x_A)(y_B - y_A) - (y_Q - y_A)(x_B - x_A),$$

obținem:

Dacă $E>0$ atunci $s(Q,A,B)=1$, dacă $E=0$ atunci $S(Q,A,B)=0$, iar dacă $E<0$ atunci $s(Q,A,B)=-1$. Observăm că pentru evaluarea funcției s sunt necesare cinci operații de scădere, două operații de înmulțire și cel mult două comparații cu zero.

b) Funcția p

Fie $A(x_A,y_A)$, $B(x_B,y_B)$, $C(x_C,y_C)$ trei puncte din planul xOy. Ne interesează dacă trecerea de la punctul A la punctul C prin intermediul punctului B se realizează în sens trigonometric sau nu (când punctele nu sunt coliniare), respectiv dacă C este între A și B (când punctele sunt coliniare). Definim funcția p astfel (fig. 2): $p(A,B,C)=-1$ dacă trecerea de la A la C prin B se face în sens invers trigonometric (în sensul acelor de ceasornic) sau A este între B și C (în cazul punctelor coliniare); $p(A,B,C)=0$ dacă C este între A și B (numai în cazul punctelor coliniare) și $p(A,B,C)=1$ atunci când deplasarea se face în sens trigonometric (în sens invers acelor de ceasornic) sau când B este între A și C (în cazul punctelor coliniare).

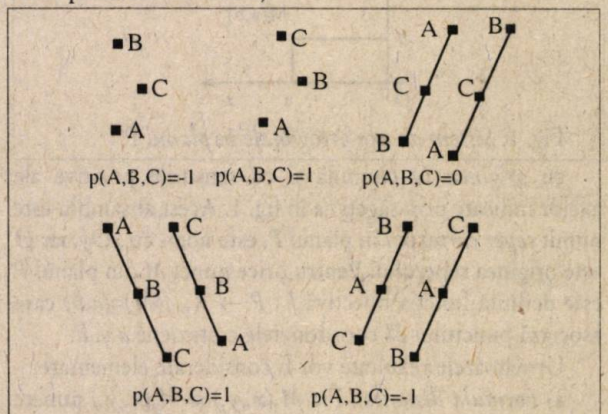


Fig 2. Configurații de puncte

În continuare vom propune câteva metode de evaluare a funcției p.

a) Metoda pantelor

Fie m_1 panta dreptei (AB) și m_2 panta dreptei (AC). Inițial presupunem că $m_1\neq m_2$. Dacă $m_1>m_2$ atunci pentru a ajunge în punctul C plecând din punctul A, trecem în deplasare invers trigonometrică prin B, deci $p(A,B,C)=-1$. Dacă $m_1<m_2$ atunci deplasarea se va face în sens trigonometric, deci $p(A,B,C)=1$. Deoarece $m_1=(y_B-y_A)/(x_B-x_A)=dy_1/dx_1$, $m_2=(y_C-y_A)/(x_C-x_A)=dy_2/dx_2$, testul de comparare între m_1 și m_2 (ce necesită două împărțiri plus o

verificare suplimentară în cazul segmentelor verticale), poate fi transformat astfel:

$m_1>m_2 \Rightarrow dy_1/dx_1>dy_2/dx_2$ Ţ $dy_1dx_2>dx_1dy_2$ Ţ $p(A,B,C)=-1$;

$m_1<m_2 \Rightarrow dy_1dx_2<dx_1dy_2 \Rightarrow p(A,B,C)=1$.

Dacă $m_1=m_2$ se va face şi o analiză a lungimii segmentelor AB şi AC. Prin urmare: $p(A,B,C)=1$ dacă $dx_1dy_2 - dy_1dx_2>0$ sau punctele sunt coliniare şi $AB<AC$; $p(A,B,C)=0$ dacă punctele sunt coliniare şi $AC<AB$, $p(A,B,C)=-1$ dacă $dx_1dy_2 - dy_1dx_2<0$ sau punctele sunt coliniare şi, fie $dx_1dx_2<0$, fie $dy_1dy_2<0$. Deoarece se poate evita calculul radicalului prin compararea pătratelor lungimilor segmentelor, pentru evaluarea funcţiei p sunt necesare maxim şapte adunări/scăderi, maxim patru înmulţiri, maxim două comparaţii şi patru operaţii de atribuire.

b) Metode trigonometrice

Se pot utiliza formule trigonometrice pentru determinarea unghiurilor dreptelor (AB) şi (AC), dar utilizarea acestora necesită timp mare de calcul. Lăsăm cititorului plăcerea de a face un studiu în această direcţie.

2. Intersecţia a două segmente de dreaptă

Formularea problemei:

Fie AB şi CD două segmente cu extremităţile specificate prin coordonate carteziene: $A(x_A,y_A)$, $B(x_B,y_B)$, $c(x_C,y_C)$, $D(x_D,y_D)$. Decideţi dacă segmentele AB şi CD au intersecţia nevidă.

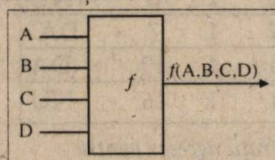


Fig. 3. Funcţia f

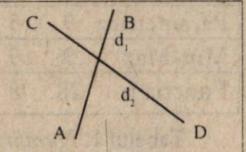


Fig. 4. Segmente intersectate

Fie f funcţia cu valori logice ce ia valoarea *true* dacă segmentele AB şi CD au intersecţia nevidă şi valoarea *false* altfel (fig. 3).

2.1. Metoda analitică

Pornim de la următoarea observaţie (fig. 4): Segmentele AB şi CD au intersecţia nevidă dacă şi numai dacă

i) A şi B sunt separate de dreapta (CD), notată d_2 şi

ii) C şi D sunt separate de dreapta suport (AB), notată d_1.

Pentru a stabili relaţiile analitice ce exprimă condiţiile i) şi ii), utilizăm ecuaţiile dreptelor d_1 şi d_2 precum şi proprietatea de separare în semiplane. Avem

$$(d_1): ux+vy+w=0, u=y_B-y_A, v=x_A-x_B, w=y_Ax_B-x_Ay_B.$$

Atunci C şi D sunt separate de dreapta (d_1) dacă şi numai dacă

$$(ux_C+vy_C+w).(ux_D+vy_D+w)<0 \qquad (1a)$$

De asemenea :

$$(d_2): u'x+v'y+w'=0, u'=y_D-y_C, v'=x_C-x_D, w'=y_Cx_D-x_Cy_D.$$

iar A şi B sunt separate de dreapta (d_2) dacă şi numai dacă

$$(u'x_A+v'y_A+w').(u'x_B+v'y_B+w')<0. \qquad (1b)$$

Deci pentru metoda analitică obţinem: $f(A,B,C,D)$ *=true* dacă relaţiile (1a) şi (1b) au loc simultan, şi $f(A,B,C,D)$*=false* în caz contrar. Numărul de operaţii necesare pentru evaluarea funcţiei f este prezentat în tabelul 1.

Observaţie:

Această metodă nu determină şi coordonatele punctului de intersecţie (în caz că există).

2.2. Metoda vectorială

Vom folosi noţiunile de vector, produs scalar a doi vectori şi produs vectorial. Vom considera planul xOy şi vectorii AC, AD, BC şi BD (fig. 5). Produsele vectoriale $AC \times AD$ şi $BC \times BD$ sunt vectori perpendiculari pe planul xOy. Dacă aceştia sunt de sens opus atunci produsul lor scalar este negativ. Dacă au acelaşi sens, produsul lor scalar este pozitiv.

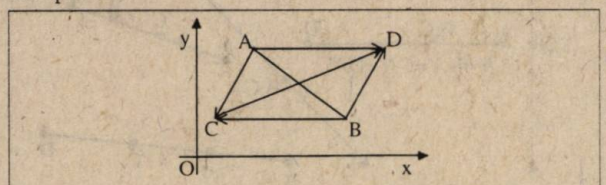


Fig. 5. Vectorii AC, AD, BC şi BD

Putem formula următoarea observaţie: *Intersecţia segmentelor AB şi CD este nevidă dacă şi numai dacă, fie produsul scalar al vectorilor $AC \times AD$ şi $BC \times BD$ este negativ, fie produsul scalar al vectorilor $CA \times CB$ şi $DA \times DB$ este negativ.*

Se ştie că într-un spaţiu ortonormat $Oxyz$ cu vectorii unitari i, j, k, produsul vectorial $AC \times AD$ se scrie astfel:

$$AC \times AD=((x_C-x_A)(y_D-y_A)-(x_D-x_A)(y_C-y_A))k.$$

Similar

$$BC \times BD=((x_C-x_B)(y_D-y_B)-(x_D-x_B)(y_C-y_B))k.$$

Atunci produsul scalar $(AC \times AD) . (BC \times BD)$, notat cu p, este egal cu:

$$p := ((x_C-x_A)(y_D-y_A)-(x_D-x_A)(y_C-y_A))$$
$$((x_C-x_B)(y_D-y_B)-(x_D-x_B)(y_C-y_B)).$$

Analog, produsul scalar $(CA \times CB).(DA \times DB)$, notat cu q, este egal cu:

$$q := ((x_A-x_C)(y_B-y_C)-(x_B-x_C)(y_A-y_C)).$$
$$((x_A-x_D)(y_B-y_D)-(x_B-x_D)(y_A-y_D)).$$

Obţinem că $f(A,B,C,D)$*=true* dacă şi numai dacă $p<0$ şi $q<0$. În caz contrar, $f(A,B,C,D)$*=false*. După prelucrarea expresiilor lui p şi q, identificarea factorilor/termenilor ce se repetă în procesul de evaluare se obţine necesarul de operaţii aritmetico-logice din tabelul 1.

2.3. Metoda parametrică

În continuare vor fi utilizate ecuaţiile parametrice ale unei drepte ce trece prin două puncte (vezi secţiunea 1.1, g):

$$(d_1)\ x=x_A+(x_B-x_A)u;\ y=y_A+(y_B-y_A)u, u \in R;$$

$(d_2)\ \ x = x_C + (x_D - x_C)v;\ y = y_C + (y_D - y_C)v, v \in R.$

Atunci segmentul AB (respectiv CD) se obține pentru $u \in [0,1]$ (respectiv $v \in [0,1]$). Dacă cele două segmente se intersectează, atunci dreptele suport d_1 și d_2 au intersecția nevidă. Deci, sistemul cu două ecuații liniare în necunoscutele u și v are soluțiile:

$$u = \frac{(x_C - x_A)(y_C - y_D) - (x_C - x_D)(y_C - y_A)}{(x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)},$$

$$v = \frac{(x_B - x_A)(y_C - y_A) - (x_C - x_A)(y_B - y_A)}{(x_B - x_A)(y_C - y_D) - (x_C - x_D)(y_B - y_A)}$$

Obținem că: $f(A,B,C,D)=true$ dacă și numai dacă $u \in [0,1]$ și $v \in [0,1]$ simultan. Altfel $f(A,B,C,D)=false$. Complexitatea evaluării funcției f este prezentată în tabelul 1.

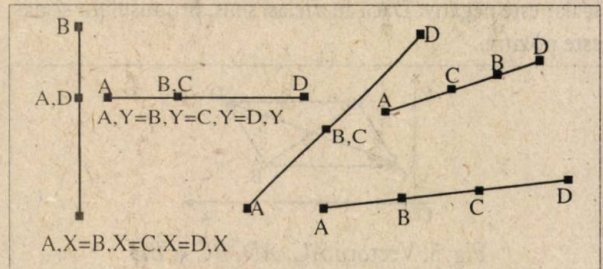


Fig. 6. Cazuri particulare

Observații:

1. Este necesar un test preliminar privind paralelismul dreptelor suport. Aceasta echivalează cu verificarea egalității:

$$(x_C - x_D)(y_B - y_A) = (x_B - x_A)(y_C - y_D).$$

Totuși, dacă punctele sunt coliniare, iar segmentele au cel puțin un punct comun, acest aspect trebuie tratat separat (fig. 6).

2. Dacă segmentele se intersectează atunci se obțin foarte ușor și coordonatele punctului de intersecție.

2.4. Metoda Min-Max

Dacă dreptele suport (AB) și (CD) se intersectează, atunci segmentele AB și CD au un punct comun dacă punctul de intersecție se află în interiorul intersecției dreptunghiurilor de diagonale AB și CD (fig. 7). Folosim notațiile prezentate în cadrul metodei analitice (secțiunea 2.1). Dacă dreptele suport au intersecția nevidă atunci coordonatele punctului de intersecție sunt:

$$x_I = \frac{vw' - v'w}{uv' - u'v},\ y_I = \frac{wu' - w'u}{uv' - u'v}$$

Atunci cele două segmente se intersectează (într-un singur punct) dacă și numai dacă:

$$max(min(x_A,x_B), min(x_C,x_D)) \le x_I \le min(max(x_A,x_B), max(x_C,x_D))\ (x\_int)$$

și

$$max(min(y_A,y_B), min(y_C,y_D)) \le y_I \le min(max(y_A,y_B), max(y_C,y_D)).(y\_int)$$

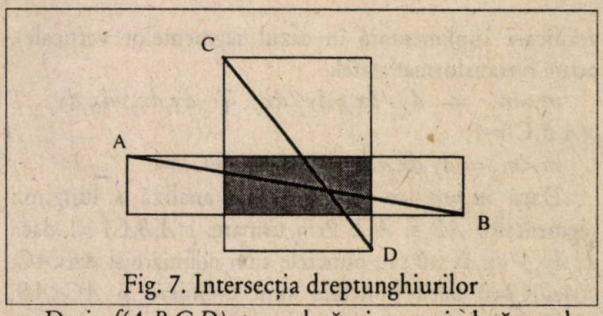


Fig. 7. Intersecția dreptunghiurilor

Deci, $f(A,B,C,D)=true$ dacă și numai dacă au loc simultan relațiile *(x_int)* și *(y_int)*. În caz contrar, $f(A,B,C,D)=false$.

Complexitatea evaluării funcției f prin această metodă este prezentată în tabelul 1.

2.5. Metoda funcției p

Utilizând funcția p (secțiunea 1.2), putem defini funcția f astfel: $f(A,B,C,D)=true$ dacă și numai dacă

$$P(A,B,C) * p(A,B,D) \le 0 \text{ și}$$

$$p(C,D,A) * p(C,D,B) \le 0 \text{ simultan.}$$

Altfel, considerăm $f(A,B,C,D)=false$. Numărul de operații necesare evaluării funcției f prin această metodă este prezentat în tabelul 1. Să observăm, că prin metoda funcției p nu se determină punctul de intersecție, dacă acesta există.

Metoda	+ (-)	*	/	<	and sau or	:=	Punctul de inters.
Analitica	14	14	0	2	1	0	NU
Vectoriala	20	10	0	2	1	0	NU
Parametrica	9	5	2	4	1	7	DA!
Min-Max	9	9	1	16	1	19	DA
Funcția p	28	8	0	8	1	16	NU

Tabelul 1. *Numărul de operații necesar pentru evaluarea funcției f prin metodele prezentate.*

3. Intersecția poligoanelor

În general intersecția a două obiecte grafice prezintă un interes special în grafica pe calculator [1,4,5,6]. Algoritmii de decupare (clipping) la limita unei ferestre dreptunghiulare sau unei ferestre poligonale, algoritmii de vizibilitate a obiectelor descrise în spațiu se bazează pe intersecții de poligoane. În prima parte se vor considera algoritmi privind intersecția dintre un poligon convex și un dreptunghi. Se vor considera apoi, algoritmi pentru determinarea intersecției a două poligoane oarecare.

3.1. Algoritmul Liang-Barsky

Fie P un poligon oarecare și D un dreptunghi cu laturile paralele cu axele sistemului de coordonate, situat în același plan cu poligonul P. Algoritmul *Liang-Barsky* folosește reprezentarea parametrică a laturilor poligonului P și calculează intersecțiile cu laturile dreptunghiului numai atunci când este necesar. Presupunem că poligonul P este dat printr-o secvență de puncte $P_1, P_2, ..., P_n$, ce definesc

laturile poligonului. Latura i începe din punctul P_i și este orientată către punctul P_{i+1}. Ecuația parametrică (vezi secțiunea 1.1) a laturii P_iP_{i+1} este:

$$R.X(t)=(1-t)\,P_i.X + tP_{i+1}.X,\ R.Y(t)=$$
$$(1-t)P_i.Y + tP_{i+1}.Y,\ 0<t\leq 1.$$

Observăm că nu se ia în considerare punctul de start al laturii. Astfel, fiecare vârf al poligonului P aparține unei singure laturi din cele două adiacente în acel vârf. Notăm cu x_i dreapta determinată de punctele P_i și P_{i+1}. Aceasta intersectează (dacă nu e paralelă cu laturile dreptunghiului) cele patru drepte suport pentru laturile dreptunghiului D. Fiecare astfel de dreaptă împarte planul celor două obiecte grafice în două semiplane. Vom numi semiplan interior semiplanul ce conține dreptunghiul D. Astfel planul este împățit în nouă regiuni (fig. 8). Regiunile sunt notate în funcție de numărul de semiplane interioare în care sunt situate.

2	3	2
3	4	3
2	3	2

Fig. 8. Divizarea planului în regiuni

O latură care intră cu extremitatea finală într-o regiune de tip 2 determină includerea vârfului dreptunghiului în lista de ieșire a punctelor intersecției celor două obiecte. Acest vârf este numit punct de întoarcere. Totuși, dacă următoarea latură care intersectează dreptunghiul are punctul de intersecție situat pe aceeași latură care a determinat un punct de întoarcere, atunci acest vârf de întoarcere este incorect și nu trebuie inserat în lista de ieșire. O dreptă x_i diagonală intersectează laturile dreptunghiului D în puncte ce corespund valorilor t_{I1}, t_{I2}, t_{E1}, t_{E2}. Deoarece o latură diagonală x_i începe și se termină într-o regiune de tip 2, t_{I1} și t_{E2} reprezintă minimul, respectiv maximul celor patru valori. Celelalte două valori determină dacă x_i intersectează dreptunghiul sau nu. În cazul în care $t_{E1}<0$ și $t_{E2} \leq 1$, latura interesectează dreptunghiul și segmentul vizibil trebuie adăugat în lista de ieșire. Dacă latura nu intersectează dreptunghiul, dreapta care o conține începe în, trece prin și se termină într-o regiune de tip 2 (fig. 9).

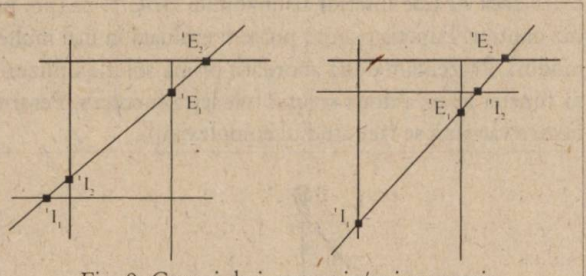


Fig. 9. Cazuri de intersecție/neintersecție

Netratând cazul particular al laturilor poligonului P, paralele cu laturile dreptunghiului D, ideile principale ale algoritmului *Liang-Barsky* sunt [7,8]:

Pentru fiecare latură x_i a poligonului P se efectuează următoarele operații:

1. Determină direcția laturii pentru a afla care este prima dreaptă a dreptunghiului intersectată de dreapta pe care se află latura x_i.
2. Determină valorile t pentru punctele de ieșire: t_{E1} și t_{E2}.
3. Dacă $t_{E2} > 0$ atunci determină t_{I1} și t_{I2}.
4. Dacă $t_{I2} > t_{E1}$ atunci
   Dacă $0 < t_{E1} \leq 1$ atunci pune vârful de întoarcere în lista de ieșire.
5. Dacă $t_{I2} \leq t_{E1}$ atunci există o parte comună:
   Dacă $0 < t_{E1}$ și $t_{I2} \leq 1$ atunci
   Dacă $t_{I2} \geq 0$ atunci pune, în lista de ieșire, punctul de intersecție;
   Dacă $t_{I2} < 0$ atunci pune, la ieșire, punctul de start;
   Dacă $t_{E1} \leq 0$ atunci pune, la ieșire, punctul de intersecție;
   Dacă $t_{E1} > 1$ atunci pune, la ieșire, punctul final.
   Dacă $0 < t_{E2} \leq 1$ atunci pune, în lista de ieșire, punctul de întoarcere.

Observații:

1. Algoritmul are complexitate liniară, numărul de etape necesare fiind $4*n$, iar pentru fiecare etapă este necesar un număr constant de operații aritmetice.

2. Generalizarea algoritmului este dificilă, datorită mulțimii de valori t_I, t_E ce apar.

3.2. Algoritmul Sutherland-Hodgman

Fie P și Q două poligoane convexe situate în același plan. Algoritmul *Sutherland-Hodgman* [10], acceptă ca date de intrare lista vârfurilor poligonului P ce definesc laturile acestuia, precum și laturile poligonului Q. Pentru a determina intersecția celor două poligoane, pentru fiecare latură a poligonului Q se parcurge lista vârfurilor poligonului P, examinându-se relația dintre oricare două vârfuri succesive ale poligonului P și latura poligonului Q, obținându-se zero, unul sau două vârfuri (într-o listă de ieșire L). Dreapta determinată de latura curentă a poligonului Q împarte planul celor două poligoane în două semiplane. Semiplanul care conține poligonul Q va fi numit semiplanul interior.

Fie A și B două vârfuri succesive ale poligonului P. Presupunem că punctul A a fost analizat în pasul anterior. Sunt posibile patru cazuri (fig. 10). În primul caz vârfurile A și B sunt în semiplanul interior, algoritmul adaugă vârful B în lista L. În cazul al doilea, latura AB intersectează latura curntă în punctul I, Vârful A fiind în semiplanul interior, iar B în cel exterior. Algoritmul adaugă punctul de intersecție I în lista de ieșire. În cazul al treilea vârfurile A și B sunt în semiplanul exterior, caz în care algoritmul nu adaugă nimic în lista L. În cazul al patrulea, latura AB intersectează latura poligonului Q în punctul I, vârful B fiind în semiplanul interior, iar A în cel exterior. Algoritmul adaugă punctele I și B în lista de ieșire.

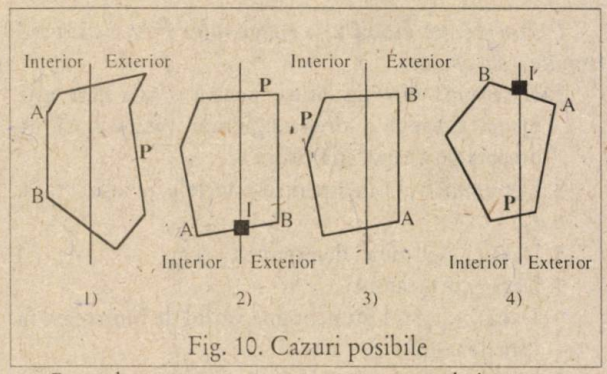


Fig. 10. Cazuri posibile

Procedura `Decupare_latura` acceptă la intrare:

`VP` — vectorul vârfurilor poligonului *P*,

`NP` — numărul vârfurilor poligonului *P*,

`x` — latura curentă a poligonului *Q*.

La ieșire se obțin:

`L` — vectorul vârfurilor obținute conform descrierii de mai sus,

`NL` — numărul vârfurilor depuse în lista `L`.

Presupunem că dispunem de o procedură eficientă, numită `intersectie` care determină punctul de intersecție dintre o latură a poligonului *P* și latura *x* precum și de funcții pentru clasificarea semiplanelor.

Pașii procedurii `Decupare_latura` sunt următorii:

```
1. NL:=0; s:=VP(NP);
2. for j:=1 to NP do
   begin
   p:=VP(j);
   if în_semiplan_interior(p)
   then
     if în_semiplan_interior(s)
     then L(NL):=p; Inc(NL)
     else i:=intersectie(s,p,x);
          L(NL):=i; Inc(NL); L(NL):=p;
Inc(NL)
   else
     if în_semiplan_exterior(s)
     then i:=intersectie(s,p,x);
          L(NL):=i; Inc(NL);
   s:=p
end;
```

Această procedură se apelează de atâtea ori câte laturi are poligonul *Q*. Pentru a descrie algoritmul pentru determinarea vârfurilor intersecției poligoanelor *P* și *Q*, notăm cu x_i latura *i* a poligonului *Q*, `V1` și `V2`, vectorii ce conțin date inițiale, intermediare sau finale privind vârfurile intersecției, cu `NV1`, respectiv `NV2` elemente.

Pașii algoritmului *Sutherland-Hodgman* sunt:

```
1. V2:=VP; NV2:=NVP;
2. Pentru fiecare latură xi a poligonului Q execută
   2.1 V1:=V2; NV1:=NV2;
   2.2 Decupare_latura(V1,NV1,xi,V2,NV2);
```

Vârfurile intersecției celor două poligoane se vor afla în tabloul *V2*, după ce s-a parcurs și ultima latură a poligonului *Q*.

Observații:

1. Dacă poligonul *Q* este un dreptunghi, atunci algoritmul *Sutherland-Hodgman* este considerat a fi un algoritm de decupare pentru vizualizarea informației grafice la nivelul unei ferestre de vizualizare.

2. Dacă poligonul *P* este concav pot rezulta laturi suplimentare pe conturul poligonului *Q*; aceste laturi false unesc poligoanele care s-ar obține în urma intersecției celor două poligoane. De aceea este necesară o postprelucrare în urma căreia sunt eliminate laturile false.

3.3. Algoritmul Weiler-Atherton

Acest algoritm [11], prelucrează lista de vârfuri furnizată de algoritmul *Sutherland-Hodgman*, după intersecția cu fiecare latură a poligonului *Q*. Algoritmul începe prin divizarea poligonului *P* față de una din laturile poligonului *Q*. Rezultatul este o listă de poligoane *LP*. Fiecare poligon din această listă este apoi divizat față de următoarea latură a poligonului *Q*, ș.a.m.d. Această prelucrare este redată în secvența următoare:

Pentru fiecare latură x a poligonului Q, pentru fiecare poligon P din lista LP execută Divide(x,P,LP). Procedura *Divide* sparge un poligon *P* din lista inițială *LP* față de latura *x* a poligonului *Q*. Dacă *P* este divizat în două sau mai multe poligoane atunci acestea sunt inserate în listă, iar poligonul inițial este eliminat.

4. Problema punctului interior

Fie *F* o figură plană a cărei frontieră este compusă din segmente. Considerăm definiția naturală a interiorului unei figuri plane. Dacă *Q* este un punct situat în același plan cu figura *F*, problema punctului interior constă în stabilirea valorii de adevăr a propoziției: *Q este un punct interior figurii F.*

În această secțiune se vor considera următoarele cazuri pentru figura F: triunghi, patrulater, poligon convex, poligon oarecare. Pentru alte aspecte se poate consulta [9].

4.1. Punct interior unui triunghi

Fie triunghiul *ABC* ale cărui vârfuri sunt $A(x_A,y_A)$, $B(x_B,y_B)$ și $C(x_C,y_C)$. Fie $M(x_M,y_M)$ un punct din planul triunghiului. Fie `pint3(A,B,C,M)` funcția care ia valoarea `true` dacă *M* este interior triunghiului *ABC* și `false`, în caz contrar. Funcția `pint3` poate fi evaluată în mai multe moduri. Prezentăm două abordări: prima soluție utilizează funcția *p*, iar a doua soluție folosește funcția *s*. Pentru fiecare variantă se face studiul complexității.

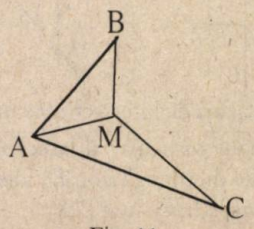


Fig. 11.

a) Metoda funcției p

În acest caz (fig. 11), funcția `pint3` poate fi definită astfel:

`pint3(A,B,C,M)=true` dacă $p(A,M,B)=1$ și $p(B,M,C)=1$ și $p(C,M,A)=1$ simultan, în caz contrar `pint3(A,B,C,M)=false`.

Luând în considerare complexitatea evaluării funcției p rezultă că pentru evaluarea funcției `pint3` prin această metodă sunt necesare 21 de operații de adunare/scădere, 12 înmulțirii, 9 teste relaționale, 2 operații logice și 12 operații de atribuire.

Observăm că punctul M aparține unei laturi a triunghiului dacă $p(A,B,M)=0$ sau $p(B,C,M)=0$ sau $p(C,A,M)=0$ cu aceeași complexitate de evaluare precum a funcției `pint3`.

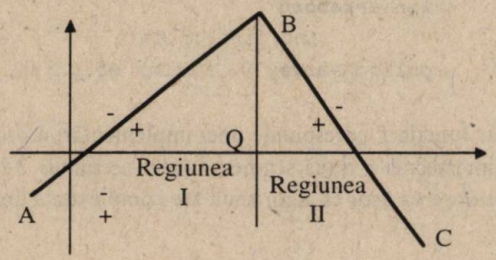


Fig. 12.

b) Metoda semnului

Vom presupune, cu păstrarea generalității, că $x_A \le x_B \le x_C$. Fie Q proiecția punctului B pe axa Ox (fig. 12). Dacă $x_M < x_B$ atunci analizăm regiunea I altfel analizăm regiunea II. Prezentăm în continuare textul pseudo-Pascal al funcției `pint3` prin această metodă [3]:

```
function pint3(A,B,C,M:point):Boolean;
var s1,s2:Real;
begin
  if (xM<xA) or (xM<xC)
  then pint3:=false
  else
    begin
      s2:=(xM-xA)*(yC-yA)-(yM-yA)*(xC-xA);
      if (xM<xB) or (xB=xC)
      then s1:=(xM-xA)*(yB-yA)-(yM-yA)*(xB-xA)
      else s1:=(yM-yC)*(xB-xC)-(xM-xC)*(yB-yC);

      pint3:=((s1≥) and (s2≤0)) or ((s1≤0)
                    and (s2≥0))
    end
end;
```

Prin această metodă sunt necesare 10 adunări/scăderi, 4 înmulțiri, 8 comparații, 4 operații logice **and/or** și maxim 3 operații de atribuire. Se observă că această metodă este mai rapidă decât metoda bazată pe utilizarea *funcției p*.

4.2. Punct interior unui patrulater simplu

Se consideră patrulaterul simplu $ABCD$ (fără autointersectări) ale cărui vârfuri sunt date prin coordonate carteziene: $A(x_A,y_A)$, $B(x_B,y_B)$, $C(x_C,y_C)$, $D(x_D,y_D)$ și $M(x_M,y_M)$ un punct din planul xOy (fig. 13).

Fie `pint4(A,B,C,D,M)` funcția ce ia valoarea `true` dacă M este punct interior patrulaterului $ABCD$ și `false` în caz contrar. O metodă de evaluare a funcției `pint4` presupune utilizarea funcției p, dar putem utiliza și funcția `pint3`, dacă împărțim patrulaterul în două triunghiuri prin intermediul unei diagonale.

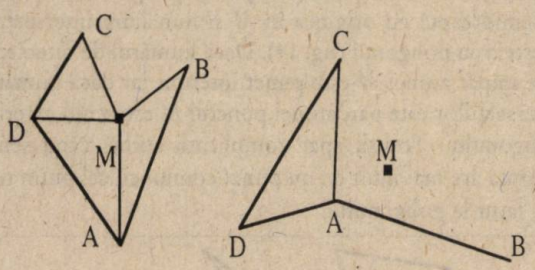


Fig. 13. Patrulatere simple
a: metoda p, b:triangularizare

a) Metoda funcției p

În acest caz (fig. 13/a), funcția `pint4` poate fi definită astfel:

`pint4(A,B,C,D,M)=true` dacă simultan au loc egalitățile:

$p(A,M,D)=1$, $p(D,M,C)=1$, $p(C,M,B)=1$
și $p(B,M,A)=1$.

În caz contar `pint4(A,B,C,D,M)=false`.

Deci evaluarea funcției `pint4` necesită 28 de operații de adunare/ scădere, 16 operații de înmulțire, 12 comparații, 3 oparații logice și 16 atribuiri; ceea ce înseamnă un efort de calcul apreciabil.

b) Metoda triangularizării

Condsiderând diagonala AC, patrulaterul $ABCD$ se împarte în două triunghiuri *(ABC și DAC)*. Deci:

`pint4(A,B,C,D,M)=pint3(A,B,C,M)`
`or pint3(D,A,C,M)`,

fiind necesare 20 adunări/scăderi, 8 înmulțiri, 16 comparații, 9 operații logice și cel mult 6 atribuiri.

4.3. Punct interior unui poligon convex

Pentru evaluarea funcției `pint(poligon,M)` se poate utiliza fie funcția p, fie metoda triangularizării (ducând diagonale sau considerând centrul de greutate și formând triunghiuri). Dacă evaluarea funcției `pint` este solicitată de mai multe ori pentru același poligon, atunci se justifică utilizarea următorului algoritm eficient[9]:

Se creează un arbore de căutare cu vârfurile poligonului convex, având drept criteriu de ordonare unghiul pe care dreapta care unește centrul de greutate și un vârf, îl

face cu axa Ox. Astfel, dacă patrulaterul are n vârfuri, se evidențiază n sectoare. Mai întâi, prin căutare în arbore, se determină sectorul în care se găsește punctul M, iar apoi se vede dacă M este în interior sau în exterior folosind, de exemplu, funcția `p` *sau funcția* `pint3` *apelată o singură dată.*

4.4. Punct interior unui poligon oarecare

Fie *pol* un poligon ale cărui vărfuri sunt *pol(i)* pentru i=1, 2, ..., n, iar laturile lui sunt segmente cu extremitățile vârfuri consecutive (presupunem că *pol*(0)=*pol*(n) și *pol* (n+1) =*pol*(1)). Fie *M* un punct din planul poligonului. Trasăm o semidreaptă cu originea în *M* și numărăm intersecțiile acesteia cu poligonul (fig. 14). Dacă numărul de intersecții este impar atunci *M* este punct interior, iar dacă numărul intersecțiilor este par, atunci punctul *M* este unul exterior poligonului. Totuși, apar complicații atunci când semidreapta are mai mult de un punct comun cu cel puțin una din laturile poligonului.

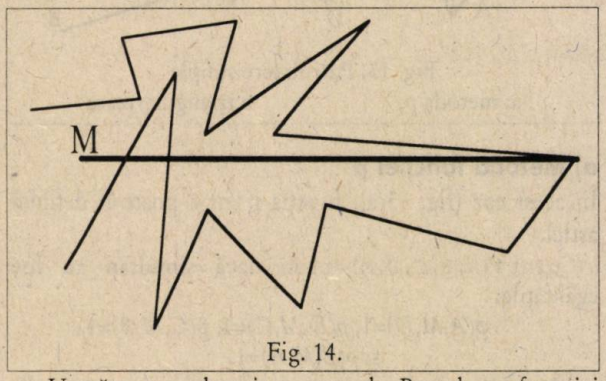


Fig. 14.

Următoarea descriere pseudo-Pascal a funcției `pint(pol,M)` utilizează o semidreaptă de test orizontală, presupune că *pol*(1) este punctul cu cea mai mică abscisă și cea mai mică ordonată pentru a ne asigura că dacă *pol*(1) este pe semidreapta de test, *pol*(0) nu poate fi pe această semidreaptă.

```
function pint(pol:poligon;M:point):Boolean;
var contor,index_curent,index_ancora:Integer;
    semidreapta,latura:line;
begin
  contor:=0; index_ancora:=0;
  pol(0):=pol(n);
  pol(n+1):=pol(1);
  semidreapta.init:=M;
  semidreapta.final:=M;
  semidreapta.final.abscisa:=maxint;
  for index_curent:=1 to n do
  begin
    latura.init:=pol(i);
    latura.final:=pol(i);
    if not f(latura,semidreapta)
    then
      begin
        latura.final:=pol(index_ancora);
        index_ancora:=index_curent;
        if    f(latura,semidreapta)    then
        Inc(contor)
      end
  end;
  pint:=(contor mod 2)=1
end;
```

În descrierea precedentă s-au utilizat următoarele tipuri de date:

```
type point=record
              x,y:tip_coordonate
           end;
     linie=record
              int,final:point;
     poligon=array(0..maxim) of point,
```

iar funcția `f` corespunde unei implementări a problemei intersecției a două segmente (vezi secțiunea 2.2). Se poate observa ușor că algoritmul are complexitate liniară.

Bibliografie:

I.C. Braid — *Designing with volumes*. Cambridge, 1974, 81-85.

F. Feito, J,C, Torres, L.A. Ureña — *Orientation, Simplicity and Inclusion Test for Planar Polygons*, Computer & Graphics, 19(4), 1995

M. Garcia, R. Segura, L.A. Ureña — *Un nuevo algorithmo de Inclusion de puntos en Triangulos*. Departamento de Informatica, Universidad de Jaen, 1997

W.K. Giloi — *Interactive Computer Graphics: Data Structures, Algorithms, Languages*, Prentice-Hall, 1978

E. Haines — *Point in Polygon Strategies*, Graphic GEMs, Academic Press, 1994

S.R. Levine — *Interactive 3-D Motion Graphics with Large Data Bases*, SLAC Report 192, Mass, 1976, 64-65

Y.D. Liang, B. Barsky — *An Analysis and Algorithm for Polygon Clipping*, CACM 26, 1983, 868-877

Y.D. Liang, B. Barsky — *A new Concept and Method for Line clipping*, ACM TOG, 3(1), 1984, 1-22

F.P. Preparata, M. I. Shamos — *Computational Geometry: An introduction*, Springer-Verlag, 1985

I.E. Sutherland, G.W. Hodgman — *Reentrant Polygon Clipping*, CACM, 17, 1974, 32-42

K. Weiler, P. Atherton — *Hidden Surface Removal Using Polygon Area Sorting*, Computer Graphics, 11(2), 1977, 214-222

Dl. Grigore Albeanu este profesor la Catedra de Informatică din cadrul Facultății de Matematică a Universității București.

E-mail: albeanu@math.math.unibuc.ro

Ești pregătit pentru mâine? **E timpul...**

Abonează-te și vei fi!

O PUBLICAȚIE COMPUTER PRESS AGORA

VIITORUL TEHNOLOGIEI INFORMAȚIEI ASTĂZI

- tehnologii Internet, intranet si Java
- tehnologii de stocare și administrarea datelor
- articole de analiză și prezentare de tehnologii fundamentale
- prezentarea unei tehnologii actuale de mare viitor
- interviuri realizate cu personalități ale domeniului
- servere de aplicații profesionale si tehnologii de vârf pentru rețele LAN si WAN

http://www.byte.ro/

+20 pagini LANTIMES

Nume: ..... Prenume: .....

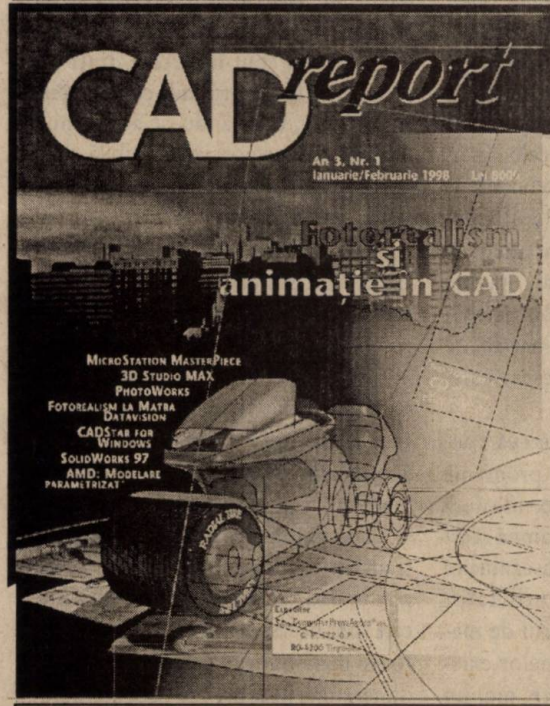
Adresă: .....

Cod poștal: ..... Localitate: ..... Tel./Fax: .....

V-am expediat ..... lei, cu mandatul poștal/ordinul de plată nr. ..... din data de ..../..../....., pentru Computer Press AGORA, în contul: 2021400027100178, Banca Transilvania, filiala Mureș.

Doresc să mă abonez la: ☐ Byte - 144.000 lei/ 1 an
- 108.000 lei/ 1 an (pentru studenți și elevi)

Expediați acest talon pe adresa: Computer Press AGORA, CP 172/1, 4300 Tg.Mureș, sau prin fax la: 065-166290. Informații suplimentare la tel: 065/166516.



Ești pregătit pentru mâine? **E timpul...**

Abonează-te și vei fi!

O PUBLICAȚIE COMPUTER PRESS AGORA

Revista care te ajută să devii performant ÎN PROIECTAREA ASISTATĂ DE CALCULATOR

COVER STORY
ANCHETĂ
PREZENTĂRI
PRACTICĂ
OPINII

http://www.cadreport.ro/

Nume: ..... Prenume: .....

Adresă: .....

Cod poștal: ..... Localitate: ..... Tel./Fax: .....

V-am expediat ..... lei, cu mandatul poștal/ordinul de plată nr. ..... din data de ..../..../....., pentru Computer Press AGORA, în contul: 2021400027100178, Banca Transilvania, filiala Mureș.

Doresc să mă abonez la: ☐ CADreport - 38.400 lei/6 apariții
- 28.800 lei/ 6 apariții (pentru studenți și elevi)

Expediați acest talon pe adresa: Computer Press AGORA, CP 172/1, 4300 Tg.Mureș, sau prin fax la: 065-166290. Informații suplimentare la tel: 065/166516.

Introducere în LINUX

de Cristian Mateescu

Interesul imens de care se bucură sistemul Linux în lumea academică și, din ce în ce mai mult, în mediile comerciale, ne-a determinat să inițiem o serie de articole care-i vor fi dedicate. Această primă prezentare este menită să vă dezvăluie câteva facilități și să vă familiarizeze cu terminologia. Următoarele vor detalia aspectele fundamentale.

Ce este Linux-ul? Sunt sigur că mulți dintre voi și-au pus această întrebare. Linux-ul este un sistem de operare de tip Unix, distribuit gratuit, compatibil cu standardul POSIX, dezvoltat în mediu academic în Statele Unite. În articolul prezent voi încerca să fac o scurtă prezentare a acestui sistem de operare, a principalelor facilități oferite de acesta precum și o trecere în revistă a necesarului de hardware și a echipamentelor suportate.

Istoria Linux-ului începe către sfârșitul anului 1991, când un student finlandez, Linus Torvalds, se gândește să-și scrie propriul sistem de operare multitasking pentru mașina sa Intel386. Nu voi insista prea mult asupra acestui subiect fiindcă povestea acestui sistem de operare a fost prezentată într-un număr precedent al revistei. Ideea este că, pornind de la acest proiect, Linux-ul s-a dezvoltat enorm, beneficiind de contribuțiile aduse de sute sau chiar mii de programatori din toată lumea, ajungând ca astăzi să fie un sistem de operare matur, stabil, ce oferă facilități comparabile cu cele găsite în sisteme comerciale Unix.

Facilități

Probabil facilitățile cele mai spectaculoase aduse de Linux sunt faptul că este un sistem de operare multitasking și multiuser. Ce înseamnă acest lucru? Păi, să le luăm pe rând.

Multitasking înseamnă posibilitatea de a rula mai multe programe (procese) în același timp, fără ca vreunul dintre ele să fie incomodat de celălalt.

Multiuser înseamnă facilitatea ca doi sau mai mulți utilizatori să ruleze aplicații simultan pe același calculator. Un utilizator care fie se află la un terminal conectat la calculator, fie îl accesează de la distanță din orice punct de pe glob, poate efectua aceleași operații ca și cum s-ar afla la consola acestui calculator, toate acestea în paralel cu ceilalți utilizatori care accesează sistemul respectiv.

Pentru a proteja utilizatorii care accesează și rulează simultan programe pe același calculator, Linux-ul oferă **protecția memoriei**. Ce înseamnă acest lucru? Protecția memoriei înseamnă că fiecare program al unui utilizator rulează în spațiul său propriu de memorie și datorită mecanismelor existente în sistemul de operare, nici un utilizator nu poate accesa spațiul de memorie folosit de un alt utilizator. De asemenea, nici un utilizator nu poate accesa spațiul de memorie folosit de sistemul de operare, în așa fel încât un program prost scris sau rău intenționat nu poate bloca întreg sistemul de operare, ci numai sesiunea aferentă acestuia.

O altă facilitate interesantă a Linuxului este **încărcarea la cerere** (*load on demand*). Aceasta înseamnă că la un moment dat sunt încărcate dintr-un program numai porțiunile ce sunt efectiv folosite, utilizându-se astfel mult mai eficient memoria disponibilă. Sunt oferite de asemenea biblioteci partajate statice și dinamice.

O mare parte din funcționalitatea Linux-ului este dată de memoria sa virtuală paginată. Memoria virtuală este compusă din memoria fizică existentă în sistem și din spațiul de swap aflat pe o partiție dedicată exclusiv acestui scop și este alcătuită din pagini de 4 Kb. Mecanismul de paginare este optimizat, în sensul că atunci când un proces necesită mai multă memorie fizică, nu este transferat pe disc un alt proces complet ci doar atâtea pagini de memorie câte sunt necesare noului proces. Se pot folosi maximum 16 partiții de swap de dimensiune maximă 128 Mb, oferindu-se astfel un spațiu de swap de maximum 2 Gb.

Compatibilitate

Linux-ul este compatibil la nivel de surse cu standardele POSIX, System

V și BSD. Un alt mare avantaj al Linux-ului este că are disponibile sursele complete, inclusiv cele ale nucleului, driverelor, uneltelor de dezvoltare precum și ale programelor utilizator. Oricine poate studia și modifica aceste surse după bunul plac, acest lucru constituind un foarte bun mijloc de învățare. Există desigur și aplicații comerciale ale căror surse nu sunt disponibile, însă toate sursele sistemului de operare, precum și cele ale programelor distribuite sub licență GPL (GNU Public License) sunt gratuite. Vă veți bucura probabil să aflați că aplicații cu care sunteți obișnuiți pe platforma Windows, precum Netscape Communicator, Adobe Acrobat Reader, Corel Word Perfect și multe altele sunt disponibile pentru Linux, iar din categoriile celor care nu au fost portate, există multe aplicații similare, gratuite sau comerciale, de unde puteți alege.

Linux-ul este un sistem de operare foarte deschis, în sensul că poate avea acces la o mare varietate de sisteme de fișiere. Probabil cel mai interesant de știut este că pot fi accesate partiții de tip DOS și VFAT (partiții Windows95/NT ce folosesc nume lungi de fișiere). Există de asemenea un driver experimental ce permite accesul la sisteme de fișiere de tip NTFS4, dar numai pentru citire (read-only), precum și un petic (patch) ce permite accesul la partiții DOS comprimate. Sunt suportate de asemenea sistemul de fișiere HPFS-2 al OS/2 (read-only), sistemul HFS disponibil pe Macintosh precum și sistemele de fișiere uzuale disponibile pe CD-ROM (de exemplu ISO-9660).

Foarte bine stă Linux-ul, de asemenea, la partea de rețele și protocoale de comunicație. Acesta suportă toată gama de protocoale TCP/IP, incluzând ftp, telnet, NFS și multe altele.

În noile nuclee experimentale care se află într-o dezvoltare continuă, sunt suportate și protocoale precum Ipv6, IPX, NetBEUI, Netrom, AX.25 și altele. Linux-ul oferă un server numit Samba, ce poate exporta discuri pe care stațiile Windows95/NT le văd ca și cum ar proveni de pe calculatoare rulând gama de produse DOS/ Windows, precum și un emulator de server Novell NetWare, numit Mars-NEW.

Sisteme de operare

DOS, Windows NT, Linux, Unix, MacOS... Ce sunt toate acestea? Nimic altceva decât sisteme de operare. Dar ce este de fapt un sistem de operare?

Să ne imaginam controller-ul de floppy aflat într-un PC obișnuit. Pentru controlul acestuia există 16 comenzi, fiecare specificată prin încărcarea unui număr între 1 și 9 octeți într-un registru de periferic. Comenzile se referă la citirea și scrierea datelor, mutarea brațului discului, formatarea pistelor, inițializarea, detectarea, resetarea și recalibrarea controller-ului și a discurilor. Fiecare dintre aceste comenzi necesită 13 parametri împachetați în 9 octeți. Când o operație este terminată, controller-ul returnează 23 de indicatori de stare împachetați în 7 octeți.

Vi se pare complicat? Nici o problemă. Aici intervine sistemul de operare. Acesta realizează interfața dintre dispozitivele periferice (floppy, hard disc, CD-ROM, placă video, tastatură, mouse etc.) în așa fel încât să nu mai avem de-a face cu astfel de comenzi complexe ca mai sus. Deci, unul din rolurile cele mai importante ale sistemului de operare este de a ne feri de modul foarte greoi și complicat de a accesa și controla dispozitivele periferice și de intrare/ieșire, precum și de a ne oferi o interfață (un set de instrucțiuni) mai simplă.

Un alt rol al sistemului de operare este de a defini structura sistemului de fișiere folosit, precum și a drepturilor de acces la acesta.

Un alt rol, la fel de important, al sistemului de operare este acela de gestiune a resurselor. Să ne imaginăm, spre exemplu, ce s-ar putea întâmpla pe un sistem pe care trei utilizatori ce lucrează simultan încearcă să trimită spre imprimantă câte un document. SO-ul se ocupă de programarea acestor sarcini (task-uri), pentru a nu apărea fragmente suprapuse din fiecare document.

Pentru a rezuma, sistemul de operare este un set de programe ce se încarcă la pornirea calculatorului și realizează interfața dintre utilizator și periferice (dispozitivele de intrare/ieșire), definește structura sistemului de fișiere folosit și gestionează (asigură partajarea) resursele calculatorului.

Software

Pentru Linux există soft disponibil cam în orice domeniu imaginabil.

Voi încerca să fac în continuare o scurtă enumerare (departe de a fi completă) a programelor și categoriilor de programe disponibile pentru Linux:

- Utilitare standard oferite de Unix: ls, tr, sed, awk și multe altele;
- Unelte de dezvoltare software: gcc, gdb, make, bison, flex, perl, rcs, cvs, prof;
- Limbaje de programare: C, C++, Objective C, Java, Modula-3, Modula-2, Oberon, Ada95, Pascal, Fortran, ML, scheme, Tcl/tk, Perl, Python, Common Lisp;
- Medii grafice: X11R5 (XFree86 2.x),

X11R6 (XFree86 3.x), MGR, MetroX, Accelerated-X;
- Editoare: GNU Emacs, XEmacs, MicroEmacs, jove, ez, epoch, elvis (GNU vi), vim, vile, joe, pico, jed, și multe altele;
- Telecomunicații: Taylor UUCP, SLIP, CSLIP, PPP, kermit, szrz, minicom, pcomm, xcomm, term, Seyon precum și câteva programe de fax și voice-mail;
- Servicii de news și e-mail: C-news, innd, trn, nn, tin, smail, elm, mh, pine (pe partea de mail se suportă UUCP, POP2/3 și IMAP4);
- Procesoare de texte: TeX, groff, doc, ez, LyX, Lout, Linuxdoc-SGML;
- Jocuri: Nethack, Mud, DOOM, mai multe variante de Tetris, șah, Go și multe, multe altele;
- Multe altele care nu au avut loc în această sumară enumerare.

Licențe și distribuții

Probabil că mulți dintre voi se întreabă cum stă Linux-ul din punct de vedere al legalității. Linux-ul este distribuit gratuit sub licență GNU GPL (GNU General Public License), care uneori mai este întâlnită și sub denumirea de "copyleft". Pentru cei ce doresc să cunoască acest document, el poate fi accesat online la adresa: http://www.gnu.org/copyleft/gpl.html. Majoritatea programelor disponibile sub Linux sunt oferite sub licență GPL, deși există și unele excepții. Serverul de X-Windows este oferit sub licență MIT-X, iar unele module de rețea sub licența BSD. În general, toate programele disponibile pe site-urile FTP sunt liber distribuibile, altfel nu ar avea ce căuta acolo.

În final bănuiesc că sunteți interesați de modul cum ați putea intra în posesia unei distribuții de Linux. Există mai multe variante pe care le voi enumera:
- Ftp. Linux-ul este disponibil pentru descărcare prin ftp pe foarte multe site-uri din toate colțurile lumii. Primul loc în care ați putea încerca sunt site-urile firmelor care se ocupă de distribuții. Aceste site-uri de cele mai multe ori fiind foarte încărcate, este util să încercați un "mirror" mai apropiat. Există mai multe adrese locale de unde puteți obține Linux-ul, dintre care vă voi prezenta câteva:
  ftp://sunsite.pub.ro/pub/Linux/
  ftp://ftp.dnt.ro/pub/linux/
  ftp://ftp.kappa.ro/pub/Linux/
  ftp://ftp.logicnet.ro/pub/Linux/
- CD-ROM. Cea de-a doua și cea mai simplă variantă ar fi obținerea unui CD cu o copie a distribuției preferate. Există mai multe firme care produc astfel de CD-uri, prețul plătit nefiind pentru dreptul de utilizare al softului, ci pentru mediul de distribuție (CD) și pentru manopera de inscripționare a acestuia. Astfel de CD-uri sunt disponibile de la Red Hat Software, Walnut Creek CDROM, Infomagic și multe alte firme. Pentru detalii, consultați paginile Web:
  - http://www.redhat.com/
  - http://www.cdrom.com/
  - http://www.suse.de/
  - http://www.debian.org/

Configurații

Configurație minimală:
procesor 386;
2 MB RAM;
unitate de dischetă.

Configurație uzuală:
procesor 486/Pentium;
8/16 MB RAM;
3-400 MB spațiu pe disc.

Configurație complexă:
procesor PentiumPro/Pentium II (eventual dual);
64 MB RAM;
2-3 GB spațiu pe disc.

Cristian Mateescu poate fi contactat la adresa mateescu@ulise.cs.pub.ro, sau prin intermediul redacției din București.

Hardware suportat:

Procesoare: x86 (de la 386 în sus - Intel, AMD, Cyrix/IBM), Digital Alpha, Sun Sparc și Motorola 680x0. Este de asemenea în curs de portare pe: MIPS, PowerPC și ARM. De asemenea, suport pentru multiprocesare.

Arhitectură: magistrale ISA, VLB, PCI și EISA. Suportul pentru magistrală MCA (IBM PS/2) este încă în stadiu experimental.

Memorie: Până la 1 GB pe platformă Intel, peste platformele pe 64 de biți.

Dispozitive de stocare: discuri hard MFM, RLL, ESDI, IDE/EIDE și SCSI; benzi magnetice tip QIC și DAT; aproape toate tipurile de unități CD-ROM; discuri amovibile tip ZIP/JAZZ; etc.

Video: modurile Hercules, CGA, EGA, VGA, SVGA. Chipset-uri: Trident, S3, ET, Paradise, Hercules, ATI, Matrox, Number Nine, Diamond și altele.

Rețea: cartele ethernet și fast ethernet, de asemenea Wave Lan, FDDI și ATM, comunicații seriale, cartele multiport și ISDN.

Audio: cartele compatibile Creative Labs Sound Blaster Pro, Audio Spectrum 16, Gravis Ultrasound; de asemenea există suport MIDI.

Imprimante: aproape orice imprimantă este suportată în mod text, altele sunt suportate folosind emulatorul PostScript; este oferit de asemenea suport nativ PostScript pentru imprimante laser.

Alte echipamente: scannere, surse neîntreruptibile UPS etc.

SISTEMUL de fișiere

de Cristian Mateescu

În această a doua parte a articolului ne vom ocupa de sistemul de fișiere. Vom vedea ce este acela un sistem de fișiere, care este structura acestuia și care sunt rolurile sale principale într-un sistem de operare.

Tipuri de sisteme de fișiere

Ce este un sistem de fișiere? Ei bine, acesta se ocupă de organizarea și stocarea informației într-un mod în care aceasta poate fi ușor accesata de către sistemul de operare. Datele sunt stocate în fișiere iar acestea, la rândul lor, în directoare organizate pe mai multe nivele. Există multe tipuri de sisteme de fișiere, fiecare specific în general unui anumit sistem de operare. Spre exemplu, sistemul de fișiere FAT este specific sistemului de operare MS-DOS, VFAT32 lui Windows-ului 95 OSR2, iar NTFS este specific lui Windows NT. În mod similar, Linux-ul are un sistem propriu de fișiere (nativ) care se numește ***ext2fs*** (*extended 2-nd file system*) și oferă o mulțime de facilități.

Crearea unui sistem de fișiere

Înainte de a putea folosi un sistem de fișiere, acesta trebuie creat. Operația similară din DOS se numește formatare. Să vedem mai întâi însă cum sunt denumite sub Linux partițiile unui disc. Discurile (E)IDE care sunt conectate de obicei la controller-ul de pe placa de bază sunt denumite ***hdx***, astfel: primul este `hda`, al doilea este `hdb`, al treilea este `hdc` iar al patrulea este `hdd`. Să presupunem acum că pe primul disc avem trei partiții. Acestea vor fi denumite astfel: `hda1`, `hda2` și `hda3`. Dacă discurile nu sunt de tip (E)IDE, ci sunt de tip SCSI, ele vor fi denumite ***sdx***, astfel: primul va fi `sda`, al doilea va fi `sdb` ș.a.m.d. Și în acest caz, partițiile se vor numerota ca în cazul precedent: `sda1`, `sda2`, `sda3` etc.

Să revenim acum la problema noastră, adică la crearea unui sistem de fișiere. Pentru a crea un sistem de fișiere de tip ***ext2fs*** (specific Linux-ului) vom folosi comanda ***mke2fs*** astfel:

```
mkfs [parametri] nume_partiție
```

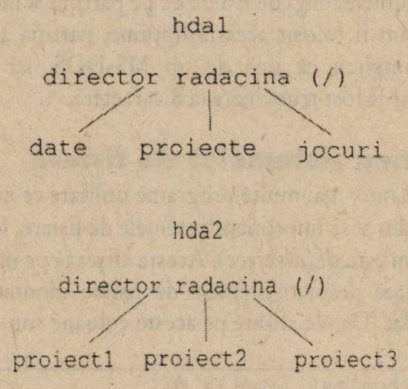
Parametrii dintre paranteze sunt opționali, iar în cazul în care aceștia nu sunt specificați sunt folosite valorile implicite. Să presupunem că vrem să cream un sistem de fișiere pe a doua partiție a primului disc. Comanda corespunzătoare va fi:

```
mkfs -t ext2fs /dev/hda2
```

Vă întrebați probabil de ce am scris `/dev/hda2` și nu direct `hda2`. Ei bine, `/dev` este un director prin intermediul căruia sistemul de operare comunică cu perifericele calculatorului. Toate aceste periferice au câte o intrare în acest director. (*dev* vine în limba engleză de la "devicè", care înseamnă dispozitiv periferic). Astfel, `-t ext2fs` este un parametru prin intermediul căruia comunicăm sistemului de operare tipul sistemului de fișiere pe care dorim să îl creăm (în cazul nostru ***ext2fs***).

Montarea și demontarea

Am creat sistemul de fișiere, însă înainte de a-l putea accesa va trebui să îl montam undeva în structura noastră de directoare. Ce înseamnă operația de montare? Să presupunem că avem o partiție principală `hda1` și o partiție secundară `hda2` cu structurile de directoare de mai jos:



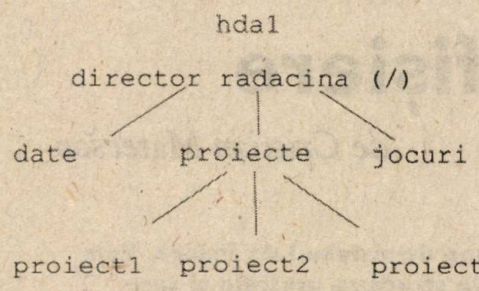
Pentru a putea accesa partiția `hda2` va trebui să o "montăm" undeva în structura de directoare a discului `hda1`. Să presupunem că vrem să o montam în directorul `/proiecte` de pe primul disc. În acest scop vom folosi comanda `mount` astfel:

```
mount /dev/hda2 /proiecte
```

Structura generală a acestei comenzi este:

```
mount partitie_de_montat director [parametri]
```

Să vedem acum cum va arata noua structură de directoare după de am "montat" partiția hda2:



Cum vom putea accesa acum directoarele de pe discul hda2? Să luam spre exemplu directorul /proiect2 de pe discul hda2. Acest director a devenit acum /proiecte/proiect2. Deci un fișier (să zicem "sumar.txt") ce exista în directorul proiecte2 pe discul hda2 acum se numește: /proiecte/proiect2/sumar.txt

Operația inversă celei de montare de numește demontare și se realizează cu ajutorul comenzii ***umount***. Sintaxa acesteia este:

```
umount partitie_de_demontat
```

De exemplu, dacă dorim să demontam partiția hda2 pe am montat-o mai devreme, vom folosi comanda:

```
umount /dev/hda2
```

De remarcat că pentru numele partiției am folosit, că și în cazul comenzii ***mount***, /dev/hda2 și nu direct hda2. Să vedem un alt exemplu. Să presupunem acum că avem o partiție ***hda1*** pe care avem instalat Windows 95 pe care dorim să o montam în directorul /win95 de pe partiția hda2 cu Linux. Vom face acest lucru astfel:

```
mount /dev/hda1 /win95 -t vfat
```

Care este rostul parametrului -t vfat? Ei bine, acesta spune sistemului de operare că vrem să montam aceasta partiție ca una de tip ***vfat***, lucru ce ne va permite să accesam numele lungi de fișiere de pe partiția Windows95. Dacă nu am fi folosit această opțiune, partiția ar fi fost montată implicit ca una de tip MS-DOS, iar numele fișierelor ar fi fost trunchiate la 8 caractere.

Gestiunea sistemelor de fișiere

Exista în Linux mai multe programe utilitare ce ne permit să gestionăm și să întreținem sistemele de fișiere. Un astfel de program este ***df*** (*diskfree*). Acesta afișează pe mai multe coloane date despre sistemele de fișiere montate la un moment dat. Datele afișate pe aceste coloane sunt:

- numele partiției;
- dimensiunea totală (în KB);
- spațiul ocupat (în KB);
- spațiul disponibil (în KB);
- spațiul ocupat (în procente);
- directorul în care a fost montată partiția.

Un exemplu de ieșire al acestei comenzi se poate vedea în *Listing 1*.

După cum se vede, avem o partiție Linux montata în directorul rădăcină (/) și alte două partiții montate în /mnt/data și respectiv /mnt/win98. Pentru a afla tipul acestor partiții avem la dispoziție o altă comandă. Putem apela comanda ***mount*** fără nici un parametru. Aceasta va afișa toate partițiile montate în momentul respectiv precum și alte informații despre acestea astfel:

- numele partiției;
- directorul în care este montata;
- tipul partiției;
- modul de acces la partiție.

Pe același sistem de mai sus, rezultatul comenzii ***mount*** este:

```
[root@galaxy /root]# mount
/dev/hda3 on / type ext2 (rw)
/dev/hda6 on /mnt/data type vfat (rw)
/dev/hda1 on /mnt/win98 type vfat (rw)
```

Se vede acum că avem o partiție Linux (ext2) montata în directorul rădăcina (/) și două partiții de tip ***vfat*** (Windows). Dreptul de acces (ultima coloana între paranteze) poate fi de două feluri:

- rw (*read/write*) - avem drept și de citire și de scriere;
- ro (*read only*) - avem drept de citire, dar nu și de scriere.

Un alt utilitar ce ne poate oferi informații utile despre sistemele de fișiere prezente (fie ele montate sau nu) este ***fdisk***. Atenție: ***fdisk*** vă poate distruge iremediabil datele de pe disc dacă este folosit necorespunzător, deci nu va recomand utilizarea lui dacă nu îi cunoașteți modul de utilizare! (vezi *Listing 2*)

Datele afișate pe coloanele sunt:

- numele partiției;
- este boot-abilă sau nu (cele ce sunt boot-abile sunt marcate cu *);
- cilindrul pe care începe partiția;
- primul cilindru al partiției;
- ultimul cilindru al partiției;
- numărul de blocuri al partiției;
- identificatorul partiției (fiecare tip de partiție are un

Listing 1

```
[root@galaxy /root]# df
Filesystem         1024-blocks      Used Available Capacity Mounted on
/dev/hda3              1064946    537825    472099      53% /
/dev/hda6              1533952   1524672      9280      99% /mnt/data
/dev/hda1               513776    403776    110000      79% /mnt/win98
```

cod asociat);
- tipul partiției (asociat codului de mai sus).

Tipuri de fișiere

După cum am văzut, datele sunt stocate în fișiere. Acestea pot fi de mai multe feluri, cele mai importante fiind:

- executabile sau fișiere de date;
- directoare;
- fișiere *device*;
- fișiere *link*.

Le vom analiza în continuare pe fiecare în parte. Fișierele "normale" sunt cele mai numeroase. Acestea sunt fie fișiere executabile, fie fișiere de date, fie fișiere de configurare etc.

Al doilea tip de fișiere sunt directoarele. Acestea nu stochează date în sensul propriu al cuvântului ci stochează fișiere sau alte directoare. Ele sunt foarte utile pentru organizarea fișierelor pe disc.

Fișierele de tip *device* sunt un tip special de fișiere ce se află în directorul `/dev`. Ele nu sunt fișiere normale, deoarece nu stochează nici un fel de date, ci sunt legături către diverse periferice și subansamble ale calculatorului. Prin intermediul acestui tip de fișiere, calculatorul comunică cu aceste periferice.

În fine, un alt tip de fișiere sunt cele de tip *link*. Acestea nu sunt fișiere propriu-zise, ci legaturi (*link*-uri) către un fișier din oricare dintre tipurile precedente. Să presupunem că avem un fișier mare în directorul `/date` și dorim să avem o copie a acestuia și în directorul `/proiect` însă nu dorim să ocupăm de două ori mai mult spațiu pe disc. În această situație vom crea în cel de-al doilea director o legătură (*link*) către primul fișier. Fișierul va apare ca fiind prezent în ambele directoare, însă va ocupa spațiu pe disc o singură dată. Mai mult, orice modificare efectuată asupra fișierului va fi reflectată de către toate legăturile acestuia în mod automat.

Drepturi de acces

Pentru a studia drepturile de acces la fișiere vom executa comanda `ls -al` (echivalentul comenzii *dir* din DOS) într-un director (*Listing 3*).

Prima coloană a acestui listing reprezintă drepturile de acces la fișiere și este compusă din 10 câmpuri: `drwxrwxrwx`. Aceste drepturi sunt interpretate astfel:

- prima literă poate fi:
  "d" - director;
  "l" - legătură;
  "-" - fișier ordinar.
- primul grup `rwx`: drepturile de acces ale proprietarului fișierului:
  "r" - drept de citire;
  "w" - drept de scriere;
  "x" - drept de execuție.

Dacă vreuna dintre aceste litere este prezentă înseamnă că acel drept există, dacă acea litera este înlocuită cu un "-" înseamnă că acel drept lipsește.

- al doilea grup `rwx`: drepturile de acces ale grupului de utilizatori din care face parte proprietarul fișierului (similar cu cele de mai sus).
- al treilea grup `rwx`: drepturile de acces ale celorlalți utilizatori (similar cu cele de mai sus).

Cea de-a doua coloană conține numărul de legaturi existente către acel fișier.

Următoarele două coloane reprezintă proprietarul fișierului și grupul de utilizatori din care face parte acesta.

Cea de-a cincia coloana specifică dimensiunea fișierului în biți iar următoarele două data și ora la care a fost modificat ultima dată fișierul.

În fine, ultima coloana o reprezintă numele fișierului.

Structura de directoare

Sistemul de directoare al Linux-ului are o structura destul de complexă. În cele ce urmează voi încerca să o descriu pe

Listing 2

```
[root@galaxy   /root]# fdisk
Using          /dev/hda as default device!

Command (m for help): p

Disk /dev/hda: 255 heads, 63 sectors, 784 cylinders
Units=cylinders of 16065 * 512 bytes

Device    Boot     Begin      Start        End     Blocks        Id     System
/dev/hda1    *         1          1         64     514048+        6     DOS 16-bit >=32M
/dev/hda2              65         65        638    4610655        5     Extended
/dev/hda3             639        639        775    1100452+      83     Linux native
/dev/hda4             776        776        784     72292+       82     Linux swap
/dev/hda5              65         65        447    3076416        7     OS/2 HPFS
/dev/hda6             448        448        638    1534176        6     DOS 16-bit >=32M

Command (m for help): q
```

scurt și să explic rolul principalelor directoare din această structură.

Directorul de bază al acestei structuri este directorul rădăcină (/). Acesta nu conține fișiere ci doar directoarele principale ale Linux-ului. Structura acestuia este următoarea:

```
/------/bin
     |--/boot
     |--/dev
     |--/etc
     |--/home
     |--/lib
     |--/lost+found
     |--/misc
     |--/mnt
     |--/proc
     |--/root
     |--/sbin
     |--/tmp
     |--/usr
     \--/var
```

• **Directorul /bin.** Conține utilitare sistem și nu are subdirectoare.

• **Directorul /boot.** Conține fișierele necesare boot-ării sistemului (inclusiv kernel-ul) și nu are subdirectoare.

• **Directorul /dev.** Este un director special ce conține legături către toate perifericele și subansamblele calculatorului. Are un subdirector `/dev/inet` ce conține link-uri către protocoalele de Internet.

• **Directorul /etc.** Este un director foarte important ce conține fișiere de configurare ale sistemului. Structura să este următoarea:

```
/etc---/X11
     |--/cron.daily
     |--/cron.hourly
     |--/cron.monthly
     |--/cron.weekly
     |--/default
     |--/httpd
     |--/logrotate.d
     |--/mail
     |--/news
     |--/pam.d
     |--/pcmcia
     |--/ppp
     |--/profile.d
     |--/rc.d
     |--/security
     |--/skel
     |--/slip
     |--/sysconfig
     |--/network-scripts
     |--/uucp
     \--/vga
```

Cele mai importante subdirectoare ale acestuia sunt: `/etc/X11` (fișierele de configurare pentru mediul grafic X Windows), `/etc/httpd` (fișierele de configurare pentru serverul de Web Apache), `/etc/ppp` (fișierele de configurare pentru protocolul de rețea ppp), `/etc/rc.d` (fișierele de inițializare ale sistemului, specifice fiecărui *runlevel*) și `/etc/network-scripts` (scripturi de configurare a rețelei).

• **Directorul /home.** Este folosit pentru stocarea

Listing 3

```
[root@galaxy /root]# ls -al
total 55
drwxr-xr-x          19 root         root         1024   Jul 28 15:22 .
drwxr-xr-x          19 root         root         1024   Jul 28 15:22 ..
drwxr-xr-x           2 root         root         1024   Oct 30   1997 .automount
drwxr-xr-x           2 root         root         2048   Jul 7 01:03 bin
drwxr-xr-x           2 root         root         1024   Jul 7 02:25 boot
drwxr-xr-x           3 root         root        21504   Jul 28 12:11 dev
drwxr-xr-x          25 root         root         3072   Jul 28 12:11 etc
drwxr-xr-x           5 root         root         1024   Jul 6 22:02 home
drwxr-xr-x           4 root         root         2048   Jul 7 01:14 lib
drwxr-xr-x           2 root         root        12288   Jul 6 21:54 lost+found
drwxr-xr-x           2 root         root         1024   Oct 24 1997 misc
drwxr-xr-x           7 root         root         1024   Jul 7 14:19 mnt
dr-xr-xr-x           5 root         root            0   Jul 28 15:10 proc
drwxr-xr-x           7 root         root         1024   Jul 28 15:26 root
drwxr-xr-x           3 root         root         2048   Jul 6 22:04 sbin
drwxr-xr-x           3 root         root         1024   Jul 6 19:25 tftpboot
drwxrwxrwt           4 root         root         1024   Jul 28 15:19 tmp
drwxr-xr-x          19 root         root         1024   Jul 6 22:00 usr
drwxr-xr-x          16 root         root         1024   Jul 6 22:05 var
```

directoarelor rădăcină ale fiecărui utilizator din sistem și are câte un subdirector pentru fiecare dintre aceștia. Mai conține, de asemenea, subdirectoare pentru ftp, httpd și samba (subdirectoare în care se află fișierele de configurare precum și cele de date ale acestora).

• **Directorul /lib.** Este un director foarte important care conține bibliotecile sistemului de operare. Structura sa este:

```
/lib---/modules
     \--/security
```

• **Directorul /lost+found.** După cum spune și numele, este un director în care sunt puse diverse fișiere găsite în sistem și care nu își aveau locul în directoarele respective. Nu are subdirectoare.

• **Directorul /misc.** Este un director folosit în diverse scopuri. Nu are subdirectoare.

• **Directorul /mnt.** Acest director este folosit ca punct de montare pentru alte partiții din sistem. Conține mai multe subdirectoare, fiecare fiind un punct de montare pentru o partiție. Dintre acestea, `/mnt/floppy` este folosit pentru montarea unității de dischetă, iar `/mnt/cdrom` pentru montarea CD-ROM-ului.

• **Directorul /proc.** Este un director deosebit de important al sistemului. Nu conține fișiere propriu-zise, ci oferă o interfață prin intermediul căreia putem afla informații despre sistem și procesele ce rulează pe acesta. Structura de subdirectoare a acestuia este următoarea:

```
/proc--/net
     |--/scsi
     \--/sys
```

Acest director mai conține de asemenea câte un subdirector pentru fiecare proces care rulează pe sistem.

• **Directorul /root.** Este directorul ce conține fișierele utilizatorului *root* (administrator).

• **Directorul /sbin.** Conține utilitare de sistem. Nu are subdirectoare.

• **Directorul /tmp.** Acest director conține fișierele temporare ale sistemului.

• **Directorul /usr.** Este unul dintre cele mai importante directoare ale sistemului și, probabil, cel mai voluminos. Are o structură destul de complexă de subdirectoare:

```
/usr---/X11R6
     |--/bin
     |--/dict
     |--/doc
     |--/etc
     |--/games
     |--/i486-linux-libc5
     |--/i486-linuxaout
     |--/include
     |--/info
     |--/lib
     |--/libexec
     |--/local
     |--/man
     |--/sbin
     |--/share
     |--/src
     \--/tmp
```

Cele mai importante subdirectoare ale acestuia sunt: `/usr/X11R6` (conține sistemul grafic X Windows Release 6), `/usr/bin` (conține utilitare sistem), `/usr/doc` (conține setul de documentație Linux — HOWTOs, LDP, FAQs, documentație în format HTML), `/usr/i486-linux-libc5` și `/usr/i486-linuxaout` (conțin bibliotecile sistem pentru formatul ELF precum și pentru vechiul format *aout*), `/usr/local` (conține aplicații specifice sistemului în cauza), `/usr/man` (conține paginile de manual ale utilitarelor sistem) și `/usr/src` (conține codul sursă al nucleului și al altor aplicații instalate pe sistem).

• **Directorul /var.** Este un alt director important al sistemului. Acesta conține fișierele *log* ale sistemului fișierele de poștă electronică și *news* etc. Structura acestuia este:

```
/var---/catman
     |--/dhcpd
     |--/lib
     |--/local
     |--/lock
     |--/log
     |--/mars_new
     |--/named
     |--/nis
     |--/preserve
     |--/run
     |--/spool
     |--/tmp
     \--/yp
```

Cele mai importante subdirectoare ale acestuia sunt: `/var/log` (conține fișierele *log* ale sistemului de operare), `/usr/spool` (conține fișierele de poștă electronică și de *news*) și `/usr/tmp` (conține fișierele temporare ale sistemului).

Cristian Mateescu poate fi contactat prin intermediul redacției la mateescu@agora.ro sau la adresa cristim@freemail.fx.ro

Ambro Systems
P-ța Cipariu Nr. 9, Ap. 11, Cluj-Napoca 3400
Tel.: 064 - 194405 Fax: 064 - 194116
Monitoare VGA și SVGA color
550.000 - 700.000 lei
486/50, 16/200/Tastatură + Monitor color 15" multiscan
1.885.250 lei

INTERNET și mass-media

de Dan Iancu

După ce am privit Internetul ca mijloc de comunicare între persoane (e-mail) și ca mijloc de comunicare în grup (forumuri), am rămas datori cu prezentarea Internetului ca mijloc de informare în masă.

Internetul constituie astăzi cea mai mare arteră pe care circulă informațiile și cei mai importanți beneficiari ai acestei facilități sunt, desigur, agențiile de știri și ziarele. În conglomeratul format de mogulii informației, trebuie să facem distincție între cei care livrează informația ca atare și beneficiarii intermediari. Agențiile de știri (CNN, Associated Press, France Press, Reuters) sunt în mod evident primii dintr-o pleiadă de alți furnizori ai unei categorii de marfă care are foarte multă căutare. Se pare că agențiile de știri nu se mai mulțumesc să ajungă la beneficiarul final doar prin intermediul obișnuitelor canale mass-media, ci și-au deschis prin intermediul Internetului site-uri unde oricine poate afla câte ceva din cele ce erau pînă deunăzi familiare doar telexurilor și faxurilor.

Internetul asigură o gamă largă de servicii, text, video, audio, foto, inaugurând astfel o noua eră a manipulării informației. Dacă prezentarea nudă era apanajul strict al hârtiei ca suport perisabil, acum avem posibilitatea să optăm pentru a primi vestea pe calea rețelei atotstăpânitoare iar ziarele să ne facă serviciul comentariului. Această mutație de la informație la comentariu va instaura, cred, o nouă modalitate de selecție: aceea a criteriului de apartenență la un grup de opinie. Vom afla de la agenție fapta, după care - pentru a urmări implicațiile - vom apela la ziarul sau, de ce nu, e-zine-ul, care ni se pare apropiat de punctul nostru de vedere. Deci, în relația *one-to-many* în Internet există la ora actuală câteva grupuri distincte: agenții de știri, ziare, liste de știri (patronate de primele două sau pur și simplu independente), radiouri și televiziuni.

Câteva repere de slang

E-zine: "*Electronic Magazine*". Asemenea cuvântului deja intrat în limbajul uzual, e-mail, cam tot ce începe cu "e" se referă la...Internet. Electronic sau nu, slang-ul utilizat specifică o modalitate de acces, respectiv calculatorul, și nu neapărat o tehnologie. E-zine-ul nu este încă, din păcate, o sursă *sine qua non* a informației. Deși există suficiente mijloace de a primi știrile numai pe calculator, se pare că omul modern are nevoie de ziarul clasic. Trebuie făcută o delimitare între ziarele care apar și pe Internet de e-zine-uri. Dacă *New-York Times* sau *România liberă* au și o imagine pe web, un e-zine nu are un corespondent pe hârtie.

Real player. Televiziunea a acaparat în anii '80 o mare parte a informației și lumea este avidă de imagini. La ora actuală, Internetul poate transmite informația în orice formă, chiar și cea de reportaj filmat. Pentru a utiliza această facilitate, marile companii de software au pus la dispoziția publicului programe care facilitează accesul la acest mod de receptare. Dintre ele aș menționa programele din seria "Real" cu toate sufixele aferente: audio, video, player. Versiuni gratuite le puteți găsi la www.real.com.

De altfel și în site-ul Microsoft puteți găsi "Microsoft Media Player for x86 systems", recomandat și de CNN.

Front page. Termen nu tocmai consacrat, dar care este foarte sugestiv pentru cei care doresc ca atunci când deschid Internetul să primească în primul rând știri. Majoritatea celor care oferă știri pe Internet vă vor propune acum o modalitate de a recepta lucrurile care vă interesează în primul și-n primul rând. Exemplele sunt "Yahoo" pe adresa my.yahoo.com și CNN. Yahoo este unul, dacă nu primul, dintre "motoarele" de căutare pe Internet care oferă și posibilitatea de a ne modela un "front page" după necesitățile noastre. Bazat pe informații venite de la CNN, MSNBC, Wired, acest "front page" vă dă posibilitatea să veniți în contact cu informații din cele mai diverse domenii, pe care vi le puteți "asambla" după necesități, de la temperaturile la București până la ultimele concerte rock.

Push technology. Poate cel mai modern concept în domeniu, nu numai al ştirilor, ci şi în cel al reclamei, "tehnologia întâlnirii necăutate" - aş traduce oarecum literar - este modalitatea de a ieşi în întâmpinarea aşteptărilor, poate neexprimate, dar care se pot găsi pe o medie suficient de bine studiată a consumatorilor. În principiu, ne referim la modalitatea de a trimite în calculatorul utilizatorului date cerute, la intervale predefinite. Modelul este cumva agresiv şi ţine de o restructurare a conceptului de atitudine faţă de informaţie. Dacă în "pull technology" cel care dorea să primească trebuia să caute, în noua concepţie el trebuie doar să definească domeniul de interes şi va primi tot ce-l interesează, inclusiv ca un *background* ce va fi livrat de un server ce deţine "formula" de interes. Deci, teoretic măcar, "spune ce ţi-ai dori şi vei avea".

Pull technology. Vrei ceva? Caută! Acesta e pe scurt traducerea acestui model natural de informare. Nu tocmai în concordanţă cu firmele ce se vor mereu pe buzele tuturor, acest fel de a fi informat are avantajul de a dispune de date după bunul plac al fiecăruia. Desigur, va trebui să căutăm, dar vom dispune de timp şi de libertatea de a nu vrea nimic. Această tehnologie ne va face familiari cu căutările pe Internet şi modalităţile de a combina cuvinte-cheie.

Subscribe. Similar cu înscrierea pe listele de discuţie, termenul se foloseşte pentru o înscriere într-o listă de distribuţie, plătită sau nu, prin care abonatul îşi trimite adresa de e-mail urmând să primească în e-căsuţa menţionată un rezumat al ştirilor cu trimiteri la paginile Web ce cuprind tot articolul (vezi *Wired*) sau chiar ştirile ca atare (vezi TELEGRAMA). Unele e-zine-uri trimit fişiere arhivate în format "zip" sau "exe" care conţin întreaga revistă, cu imagini şi sunet (vezi "agora ON line").

Ce vom face ?

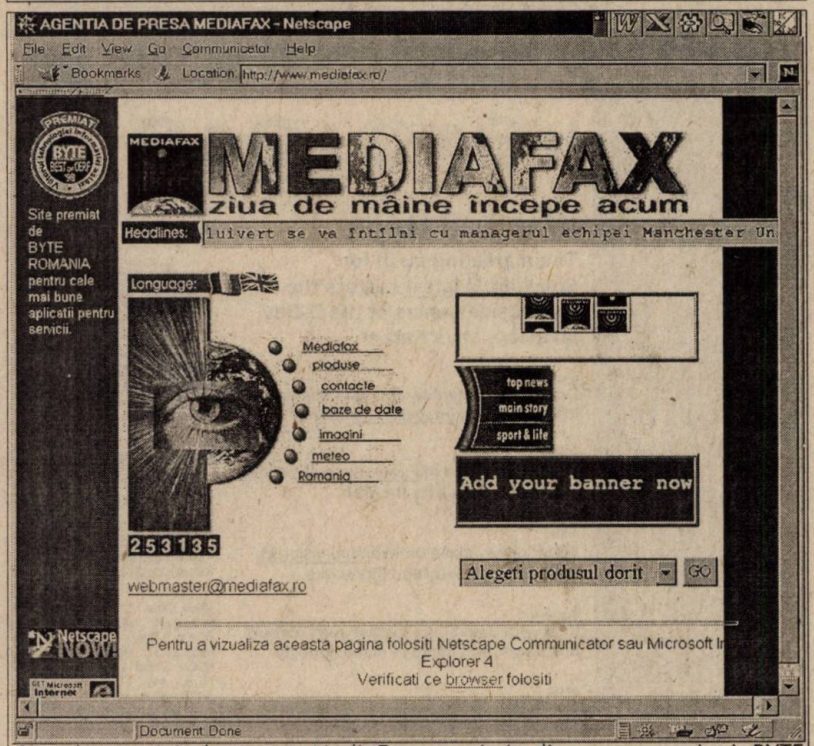
În principal, pentru a primi ştiri nu avem decât să ne "abonăm" la un canal de ştiri, să ne definim un "front page", să deschidem zilnic Internetul şi să primim calde ştirile din oricare parte a lumii. Ar mai fi câteva posibilităţi şi una dintre ele este de a face "subscribe", adică de a ne înregistra, la o listă de e-mail care trimite automat ştiri în căsuţa noastră poştală. Vă recomand, mai ales celor pasionaţi de lumea Internetului, site-ul www.wired.com unde, printr-o simplă accesare, puteţi să vă abonaţi la cea mai importantă sursă de informaţii în limba engleză. Mai nou avem şi în limba română acest fel de serviciu şi vă puteţi adresa site-ului news.agora.ro.

Agenţii de presă

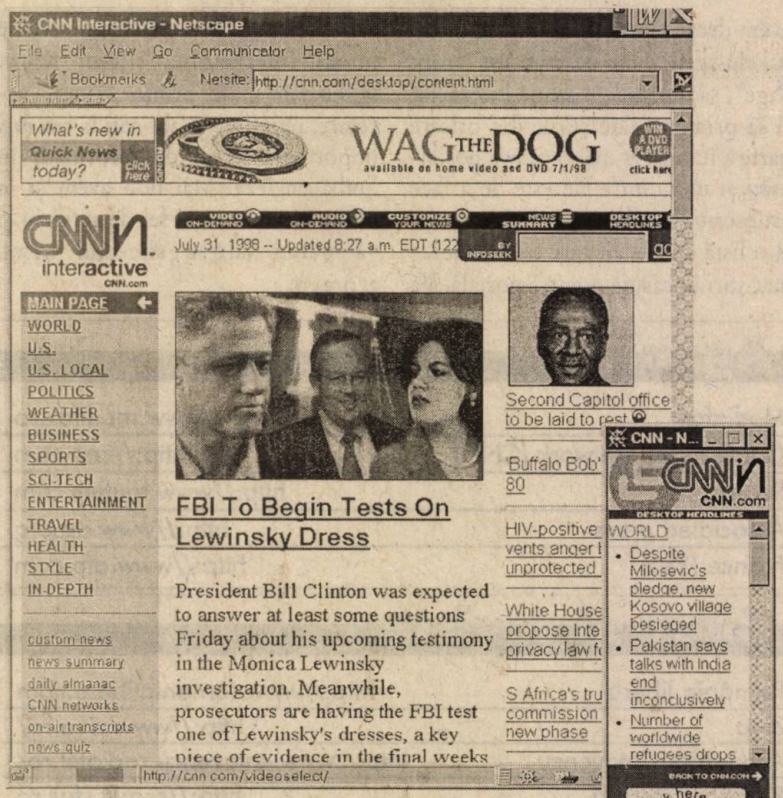
Mediafax	http://www.medifax.ro
Cable News Network (CNN)	http://cnn.com
Reuters	http://www.reuters.com
Associated press	http://www.ap.org
France Press	http://www.afp.com

Ziare şi reviste

România libera	http://www.romanialibera.com
Ziua	http://www.ziua.ro/
Academia Caţavencu	http://www.catavencu.radiotel.ro/
România literară	http://romlit.sfos.ro
22	http://www.dntb.ro/22/
New-York Times	http://www.nyt.com
Time	http://www.the-times.co.uk
The Economist	http://www.economist.com



Site-ul agenţiei de presă MediaFax a primit din partea revistei BYTE România distincţia BEST OF CERF 98 pentru cele mai bune servicii oferite de un site Web românesc.



CNN, liderul știrilor TV, atacă și piața știrilor din Internet. Site-ul său oferă servicii pe care utilizatorul le poate personaliza în funcție de domeniile de interes.

HotWired este nodul central al produselor on-line oferite de Wired Digital: Wired News, Wired Magazine, HotBot, Suck, LiveWired și altele. Tratează cu precădere implicațiile tehnologiei în viața oamenilor și a societății.

Agenții de știri

Ca volum de informații și ca prezență, este evident că agențiile de știri cuprind la ora actuală cam tot ce poate fi mai important în materie de informație nouă. Viteza actualizărilor (între 5-15 minute), aria largă de domenii acoperite, geografia foarte amănunțită pe care o dețin prin corespondenții locali, fac din marile firme surse demne de încredere și, mai ales, surse primordiale în accesul la noutăți. Fotografii, reportaje filmate, știri radio, texte - sunt modalitățile la care recurg actualmente cam toți marii "informatori". Nu este uitat nici aspectul estetic al prezentării și site-urile lor sunt modele demne de urmat prin simplitate și ușurința căutării. În plus, fiecare site deține o arhivă în care poți găsi știri relaționate precum și diverse informații utile. În România aș menționa Mediafax (www.mediafax.ro). Tot la acest capitol aș vrea să menționez și marii "provideri" de Internet, cum ar fi America Online (www.aol.com) care, în afară de serviciile pe care le oferă în mod tradițional orice furnizor de Internet, au o importantă bază de știri. Se dovedește că simpla facilitare a unui acces nu mai e suficientă, astfel încât știrile zilnice oferite (chiar dacă nu neapărat necesare pentru orice utilizator) constituie un premiu important în decizia abonatului. De altfel, AOL oferă Newsweek și New-York Times gratuit abonaților.

Ziare

Ziarele și revistele pe Internet au o prezență impresionantă, deși tirajele nu au suferit din această pricină. Diferența este că, în afară de știri, ele dau publicului și posibilitatea de a citi opinii asupra evenimentelor produse. Majoritatea ziarelor și revistelor occidentale au prezențe dintre cele mai vii pe Internet. O căutare în Yahoo Media-News vă va oferi o gamă largă de ziare și reviste. Importanța apariției pe Internet a acestor obișnuiți ai hârtiei nu este numai simpla copie electronică ci, mai ales, bazele de date oferite cu dărnicie de editori. De alt-

fel, multe dintre ziarele și revistele românești au acum un corespondent pe Internet iar în site-uri precum www.kappa.ro și www.dnt.ro se pot găsi adrese utile. Cel mai cuprinzător loc unde puteți găsi aproape toate adresele ziarelor și revistelor românești este întreținut de Tavi Ureche în Canada, la adresa www.info.plymtl.ca/tavi/arhive/ziar.

Liste de știri

Modalitatea cea mai comodă și, de fapt, cea mai ieftină, este de a te abona la un furnizor de știri, indiferent dacă el este ziar sau agenție de știri. În plus, există o serie de altfel de "știri", să le spunem "neuzuale", care fac deliciul unui public căutător de știri bizare sau care să ne descrețească fruntea. Un exemplu de astfel de site este www.suck.com, un fel de contragreutate la seriosul și importantul agent de știri în telecomunicații www.wired.com.

În principiu, modalitatea de "abonare" constă în trimiterea adresei de e-mail printr-un formular ce se găsește în pagina de web a celui care oferă acest serviciu. Mai rar trebuie să scrii la un program de tipul *listproc* sau *majordomo* pentru a te înscrie la pe o listă de distribuție. Există de altfel și modelul știrilor transmise numai prin e-mail, cum ar fi cele transmise prin buletinul "Telegrama" (http://www. yi.com/home/Miroiu Alex/romania.html) sau "Voluntar", editat de Fundația pentru Dezvoltarea Societății Civile (www.fdsc.ro).

Mâine

Internetul cumulează la ora actuală toate posibilitățile de transmitere și distribuție a știrilor, integrând televiziuni, canale de radio, presa scrisă, într-un tot unitar oferind beneficiarului posibilitatea să selecteze ce anume îl interesează. Marele avantaj este că selecția poate elimina agresiunea vizuală și, în același timp, elimină timpii pierduți în trecerea peste lucrurile care nu ne interesează. Ne putem imagina un scenariu în care fiecare membru al unei familii va găsi dimineața pe ecranul subțire ca o coală de hârtie strict ce își dorește să afle, fără să treacă incomodul ziar plin de reclame de la unul la altul.

AGORA ON LINE

Domnule redactor-sef ce este "agora ON line" ?

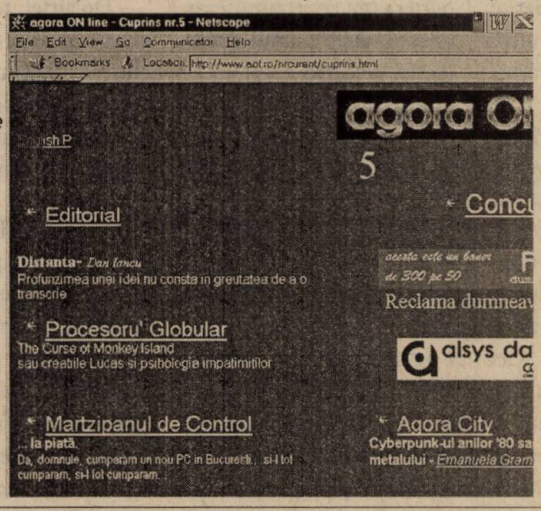
Foarte simplu spus, "agora ON line" este o revistă despre Internet, pe Internet. Încercăm să renunțăm la hârtie ca suport perisabil și finit. Mă gândesc, de fapt, la o revistă care să nu aibă un număr știut de pagini, ci doar câteva rubrici - nici ele nefiind "bătute-n cuie". Vrem să putem crea un loc în care să discutăm, noi cei care nu suntem creatori profesioniști de pagini de web, despre Internetul ca fenomen actual și despre lucruri frumoase și interesante pe care le putem găsi acolo.

Puteți să ne prezentați pe scurt rubricile acestui e-zine ?

Să începem cu concursul nostru, care oferă premii celor care în principal citesc cu atenție revista noastră. Avem un editorial care se preocupă de noțiuni importante și ne-tehnice și care exprimă, filosofic să spunem, o anumită viziune asupra Internetului. "Agora city" este o rubrică despre comunități electronice, "Martzipanul de control" una de glume, "Procesorul globular" prezintă jocuri, "Periscop" este dedicată în principal noutăților în domeniu, "Coloana infinetului" este o rubrică despre Internet în România, "De-a dura lex" ne vorbește despre conduită și legi, "Netologia" este un spațiu rezervat diverselor păreri, dar și prezentărilor de programe sau altor unelte, "Gura leului" este poșta redacției, "Don Pedrito El Duende" este piticul nostru care ne duce prin Internet prezentându-ne locuri interesante iar "WebDEX" este un dicționar pe înțelesul tuturor.

Ce aveți să le spuneți cititorilor noștri ?

Să ne scrie. Să aibă păreri și, mai ales, să folosească Internetul ca pe o minunată bibliotecă. Să nu uite că și ei sunt chemați să pună noi cărți in ea.



(Mi-am mulțumit pentru interviul pe care mi l-am luat și mi-am urat mult-mult succes).

Aaaa, Domnul reporter !

Ce s-a-ntîmplat ?

Am uitat esențialul. Adresa revistei noastre: **www.aol.ro**!!!

Dan Iancu este redactorul șef al revistei electronice "agora ON line" și poate fi contactat pe adresa diancu@agora.ro.

Mâine, în mileniul următor

Cartea de SILICIU

De Mircea Sârbu

Cartea este încă un simbol al culturii și al științei. Nici filmul, nici radioul și nici televiziunea nu i-au amenințat supremația ca principal vehicul al cunoașterii. Va rezista ea în era digitală?

Unul dintre puținele bune obiceiuri pe care mi l-am format în școală este acela de a recurge adesea la dicționare. Atunci când abordezi un subiect este bine să știi cu precizie care sunt termenii cu care operezi iar definițiile dicționarelor sunt, în cele mai multe cazuri, exacte și concise.

Ce este așadar o carte? Referința cea mai modernă a limbii române, DEX-ul, ne spune că este o „Scriere cu un anumit subiect, tipărită și legată sau broșată în volum [...]". Dacă nu suntem mulțumiți, putem încerca Dicționarul Enciclopedic, care ne dă două sensuri: „1. Reunire în volum a unui număr mai mare de 48 de file tipărite. 2. Scriere (cu conținut literar, științific, didactic etc.) a unui autor, apărută în formă de carte".

Desigur, mai am încă vreo câteva definiții la îndemână, dar cea mai potrivită pentru acest context mi se pare cea din *Le Petit Larousse*: „1. Colecție de file imprimate și reunite într-un volum legat sau broșat. 2. Volum tipărit privit din punctul de vedere al conținutului său".

Ceea ce această definiție pune cu claritate în evidență este caracterul dual al cărții: cartea ca *obiect* (coli tipărite adunate între două coperte) și cartea ca *informație* ("conținutul" pe care-l vehiculează).

Atomi și biți

Acest caracter dual este tocmai cel care determină un paradox care a însoțit întreaga evoluție a cărții: deși este un vehicul al informației, producerea cărților este dominată de aspectele legate de suportul material al acestei informații.

Într-o carte celebră (*Being Digital*, Vintage, 1995) Nicholas Negroponte, directorul fondator al MIT Media Lab, abordează această problemă în termenii ei fundamentali: atomi și biți. Atomii sunt materie iar biții sunt informație. „Deși ne aflăm, fără îndoială, într-o epocă a informației" – subliniază autorul – „cea mai mare parte a informației ne este furnizată sub formă de atomi: ziare, reviste și cărți (ca aceasta)".

De altfel, toate marile evoluții ale cărții de până acum s-au referit exclusiv la aspectul material.

De la tăblițele de lut ale sumerienilor la sulurile de papirus ale egiptenilor a fost un pas imens. Grecii și romanii au preluat invenția, perfecționând-o. Papirusul, prea fragil, a fost înlocuit cu pergamentul. Sulurile ofereau doar un «acces secvențial» la conținut, așa încât romanii le-au tăiat în bucăți dreptunghiulare și le-au prins între două coperte de protecție. Astfel a luat naștere codexul, strămoșul cărții moderne.

Până la aceasta însă mai trebuiau să treacă un mileniu și jumătate, căci abia în evul mediu europenii aveau să cunoască hârtia. Iar când, pe la jumătatea secolului XV, Johann Gutenberg inventează tiparnița cu litere mobile, premisele erau complete. Biblia editată și tipărită de Gutenberg în 1456 este prima carte modernă.

Nimic însă din această evoluție nu se referă la biți, ci doar la atomi: la suportul folosit, la structura obiectului, la tehnica producerii sale. Deși, în esență, întreaga industrie a cărții – născută ca urmare a inventării și perfecționării tiparului – nu face altceva decât să intermedieze un transfer de informație de la autor la cititor, ea operează într-o covârșitoare măsură cu bunuri din sfera materială. Hârtia, imprimarea, legarea și transportul reprezintă cea mai mare parte din costurile (și, implicit, din prețul) cărții.

Calculatoarele au pătruns în lumea cărților pe ușa din dos. Mai întâi au fost folosite în faza de pregătire pentru tipar (editarea textelor, machetare) ajungând astăzi să controleze, practic, întregul proces tehnic, până la producerea filmelor tipografice. Însă chiar și în aceste condiții, cartea a rămas – în esență – aceeași ca pe vremea lui Gutenberg. Procesarea digitală are însă o consecință importantă: vrând-nevrând, conținutul cărții trece printr-o formă digitală. Se exprimă în biți.

De aici și până la ideea de a distribui către consumatori doar biții... au rămas doar doi pași.

CD-ROM

Primul a fost posibil atunci când tehnologia discurilor optice (dezvoltate

mai ales pentru înregistrări muzicale) a fost adoptată pe scară largă în informatică. Când lectorul de CD-ROM a devenit un periferic uzual, editorii au făcut saltul în necunoscut: au început să distribuie conținutul și altfel decât pe hârtie.

În vara anului 1995 am avut pentru prima dată în mână un CD-ROM care se numea «World Literary Heritage», editat de Softbit, Inc. Spre surprinderea mea, cuprindea textul complet a nu mai puțin de 600 de cărți, aparținând unui număr de peste 150 de autori, începând cu Homer și terminând cu Winston Churchill. Biblia și Coranul coexistau cu întreaga operă dramatică și poetică a lui Shakespeare, Decameronul lui Boccacio și Discursul despre metodă a lui Descartes stăteau alături de operele Sfântul Augustin sau ale lui Omar Khayyam, totul pe un disc de plastic de câțiva centimetri. Imprimate pe hârtie, aceste cărți ar fi acoperit un perete întreg.

Mai mult, alături de operele propriu-zise se aflau și biografiile tuturor autorilor (cele mai multe însoțite și de un portret), plus înregistrări sonore ale unor actori celebrii citind fragmente din operele prezente pe CD. Mecanisme de indexare și căutare îmi permiteau să găsesc oricare dintre lucrări, după diverse criterii. Și nu doar atât: în cel mult câteva secunde (nu mai mult de 5), programul de căutare găsea toate aparițiile unor cuvinte sau grupuri de cuvinte specificate prin condiții logice oricât de sofisticate. Interfața oferea și posibilitatea de a stabili oricâte semne de carte (*bookmarks*) identificate prin nume și de a adăuga notițe. Desigur, acestea se memorau doar pe discul hard local, nefiind deci „portabile".

Impresionant, nu-i așa? Cu o singură observație: ați sta în fața calculatorului să citiți, să zicem, *Crimă și pedeapsă*? Nu cred. În schimb, ați putea folosi un astfel de CD-ROM ca o referință.

Într-adevăr, aceasta este cheia: „referință". Editorii n-au reușit să vândă literatură pe CD-ROM deoarece n-au reușit să trimită cititorii în

Microsoft Encarta 98

Microsoft Encarta este una dintre cele mai răspândite enciclopedii multimedia. Encarta 98 vine pe trei CD-uri și conține peste 32.000 de articole și circa 16.000 de elemente multimedia (imagini, clipuri audio și video, programe interactive, hărți etc.). De asemenea, un dicționar al limbii engleze (*The American Heritage Dictionary*) este disponibil în orice moment: un dublu-clic pe orice cuvânt aduce pe ecran definiția acestuia.

Alături de articolele propriu-zise, Encarta 98 mai conține și numeroase *Sidebars* (casete) cuprinzând extrase din literatură și din documente celebre, păreri ale unor personalități asupra unor chestiuni controversate, articole din presă și multe altele.

Una dintre cele mai spectaculoase facilități se referă la "călătoriile virtuale", care combină texte, imagini statice și imagini panoramice de 360 de grade, care pot fi rotite interactiv.

O altă atracție o constituie *Mind Maze*, un joc de cultură generală prin care cititorul este stimulat să găsească în enciclopedie răspunsurile la întrebările ce i se pun.

Un element mai nou îl constituie *Research Organizer*, un program care permite utilizatorului să pregătească "fișe" cuprinzând informații și citate, pe care le poate structura și organiza în diverse moduri pentru a pregăti lucrări sau prerzentări.

În fine, Encarta conține o colecție de peste 10.000 de adrese de Web, care permit extinderea în Internet, pentru mai multe amănunte, a explorărilor începute în Encarta.

Eforturile de a menține cât mai actuală informația din enciclopedie s-au concretizat într-un serviciu (*Encarta Yearbook* - almanahul Encarta) prin care utilizatorii pot descărca de pe un sit Web dedicat (http://encarta.msn.com/) actualizări lunare (articole noi sau revizuite, *Sidebars*, adrese Web noi etc.).

Microsoft Encarta 98 este disponibilă atât pentru Windows cât și pentru Macintosh.

fața computerului pentru lectură. În schimb, atunci când lucrezi și oricum ești în fața calculatorului, merită să ai referințele la îndemână. Cărțile (mai ales cele de informatică) însoțite de CD-ROM au devenit ceva obișnuit. Dar succesul cel mai notabil l-a înregistrat...

Enciclopedia pe CD

Prima enciclopedie pe CD pe care am consultat-o a fost *Compton's Interactive Encyclopedia* (editată de Compton's NewMedia, Inc.) și entuziasmul meu a fost imens. Au urmat apoi altele, fie generale fie specializate, și convingerea mea este că acestea reprezintă unul dintre principalele motive pentru care merită să ai acasă un calculator pe birou.

Cantitatea. Spre exemplu, Infopedia (editată de Future Vision Multimedia), cuprinde integral articolele din cele 29 (douăzeci și nouă!) de volume ale renumitei enciclopedii Funk & Wagnalls, dar altele se bazează pe resurse încă și mai vaste. Și

Opera deschisă

Într-o celebră lucrare apărută cu vreo 25 de ani în urmă (*Opera aperta*), celebrul scriitor și savant italian Umberto Eco emitea ideea că în receptarea operei literare apare o "complicitate" între scriitor și cititor. Dacă această abordare era una de ordin teoretic, Internetul face astăzi posibilă o "complicitate" efectivă între autor și cititor. Acesta din urmă are șansa practică de a deveni coautor, iar cartea să devină una efectiv deschisă, în lucru, într-o continuă evoluție, îmbogățindu-se mereu...

O astfel de carte (poate prima din lume) a fost pusă pe Web de firma Integrasoft, o puteți găsi la adresa http://www.java.ro și se numește *Limbajul Java*. Autorul ei, Eugen Rotariu (cu care am fost coleg de facultate), a scris-o la începutul anului 1996, când Java era mai mult o promisiune. În primăvara aceluiași an a fost editată de editura Computer Press Agora, fiind prima carte de Java în limba română. Evoluția rapidă a limbajului ar fi impus ediții succesive, care însă ar fi fost foarte greu de susținut din punct de vedere economic. În urma unei înțelegeri cu editorul, cartea a fost publicată pe Web. În curând, printr-un mecanism software realizat chiar în Java, orice cititor va putea să intervină în textul cărții, adăugând propriile observații, exemple, sfaturi, eventual corecții.

Dacă aveți posibilitatea, nu pierdeți ocazia de a o "vizita". Pe lângă conținutul ei foarte util (cuprinde și o parte introductivă foarte consistentă privind calculatoarele și programarea în general), veți putea fi părtași la un proces de creație accesibil până acum doar câtorva "aleși".

De vorbă cu autorul... inițial

Ai scris cartea "Limbajul Java" în 1996, când Java era la începuturi. Dacă ai fi scris-o acum, ar arata altfel?

E.R.: Hmm... Complet altfel. Pe vremea aceea Java era așa de nou încât era nevoie de o pledoarie în sprijinul limbajului în sine. La ora actuală limbajul este un fapt, iar ceea ce ar trebui să conțină orice carte de Java ar fi, de fapt, descrierea numeroaselor API-uri disponibile în Java. Din păcate, acesta este un domeniu extrem de dinamic. JavaSoft încearcă să recupereze cât mai repede decalajul între API-urile C/C++, care au dominat piața atâția ani, și API-urile Java. Cu toate acestea, distanța o să se mai păstreze un timp, datorită cerinței de portabilitate pentru API-urile Java.

În plus, cartea din 1996 era o prezentare semi-teoretică a unui limbaj insuficient explorat de mine. Acum, este momentul pentru a scrie cărți de Java bazate pe o experiență de câțiva ani buni. Rezultatul final ar fi foarte diferit.

Prin publicare pe Web, cartea intră oarecum în "domeniul public", deși copyright-ul rămâne. Contactul cu cititorii devine mai direct și ai putea să "compilezi" sugestiile loc și să actualizezi tu însuți conținutul din timp în timp, păstrând astfel controlul asupra conținutului. De ce nu ai ales această varianta?

E.R.: În cazul ăsta, cartea ar fi în continuare una de autor, și o singură opinie ar răzbate peste tot. Și pe urmă, cine poate hotărî al cui este copyright-ul ideilor noi, care vin ca sugestii de la altcineva? Mai mult, ar fi nevoie de o bună bucată de timp pentru a face așa ceva, iar cu timpul nu stau chiar așa de bine...

Dar cred că ideea mea are cu totul și cu totul alte origini: o perioadă de timp am predat la Universitatea din Târgu-Mureș, o universitate tânără și în căutare de identitate proprie. Problema acolo era faptul că ne lipseau "somitățile" la anumite materii pe de o parte, iar pe de alta că fiecare materie avea un singur titular, fără să existe un grup de dialog în cadrul materiei respective. Așa că studenții înghițeau obligatoriu o singură opinie, pe care trebuiau să o ia de bună. Din păcate, uneori ea nu era chiar cea mai bună... Și pe urmă, uneori am senzația că mintea noastră învață mai bine din diferențe decât din postulate... De obicei, a cunoaște două opinii înseamnă a fi îndemnat să-ți construiești propria părere despre

asta încă nu e totul. Pînă și cea modestă enciclopedie pe CD-ROM cuprinde (măcar) un dicționar de limbă, un dicționar de sinonime, un dicționar de citate, un almanah, un atlas geografic etc. Toate acestea, pe hârtie, sunt accesibile doar în marile biblioteci și foarte puțini își pot permite să le aibă în biblioteca proprie.

Accesibilitate. Dincolo de cantitatea imensă a informațiilor, enciclopediile pe CD dispun de posibilități de căutare care umilesc banalul acces în ordine lexicografică pe care edițiile clasice, pe hârtie, ni-l oferă. Imaginați-vă o căutare de genul: are legătură cu geografia, se referă la explorări și exploratori, cândva în secolul 16, e din Portugalia și vreau să aibă ilustrație. Encarta produce o listă cu două articole, exact cele pe care le căutam: Vasco da Gama și Ferdinand Magellan. Dacă nu am fi cerut imagini, am mai fi găsit șase exploratori, printre care și Diaz. Imaginați-vă o astfel de căutare printre nenumăratele volume ale Britanicii...

un anumit lucru. Este un lucru cunoscut că părerile proprii nu se uită și nu pot fi schimbate, nu-i așa?

Așa că, la universitate, visam să pot prezenta studenților un curs creat de mai multe minți, cât mai luminate, dintr-un anumit domeniu. Fiecare cu propria abordare, cu propriile exemple, cu propriul farmec.

Atâta timp cât cartea este de domeniu public, treaba asta funcționează. Fiecare își poate păstra drepturile de autor ale paragrafelor pe care le-a scris. Dacă cartea se vinde, e un pic mai complicat, dar cred că poate funcționa într-un regim similar cu o revista sau așa ceva.

Permițând cititorilor să adauge conținut, "paternitatea" cărții devine oarecum colectivă. Cum se împacă aceasta situație cu calitatea ta de autor?

E.R.: Păi, mai bine coautor la o carte bun decât autorul unei cărți slabe! Dar înclin să cred că în cazul cărților tehnice se va merge în aceeași direcție ca și în cazul software-ului: o să apară o mulțime de cărți bune gratuite. Ceea ce va trebui să plătești va fi doar suportul, adică explicațiile suplimentare, studiile pe cazuri concrete, ajutorul în caz de impas. Până la urmă, ce înseamnă să fii autorul unei cărți de programare? Să compilezi o seamă de informații venind din diverse surse și să le aranjezi după cum crezi că sunt mai ușor de digerat. O carte tehnică nu este o invenție. O carte tehnică poate, eventual, să descrie o invenție. Desigur, în cazul literaturii, afirmația nu se aplică.

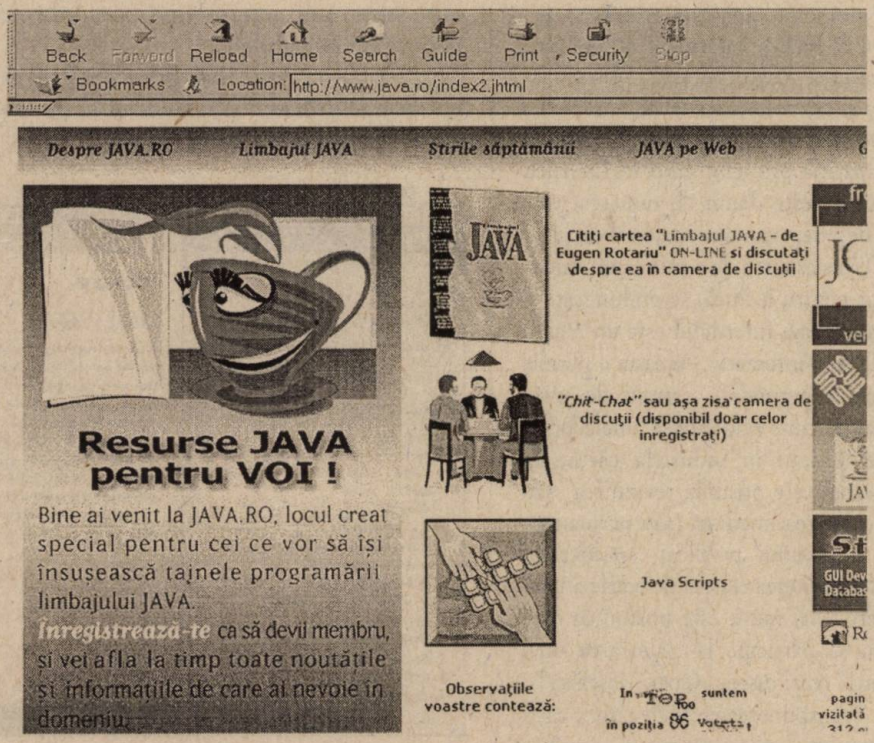
Crezi că modul de editare al cărților se va schimba esențial datorită noilor posibilități tehnologice, mai ales din domeniul calculatoarelor și Internetului?

E.R.: Da, dar nu imediat. Pe moment, există încă nostalgia cărții clasice. Pasul mare în zonă nu va fi însă modalitatea de memorare sau de lectură a cărții tehnice, ci interactivitatea. Cartea va trebui să devină un fel de "consultant digital", care să prezinte părerile celor mai bune minți într-un anumit domeniu. Până atunci, avem de a face doar cu variații pe o temă fixă.

În ceea ce privește cartea literară, cred într-adevăr că noile modalități de exprimare vor duce la apariția unei arte combinate de text, sunet și grafică animată, eventual tridimensională, care va îmbogăți dialogul dintre autor și cititor. Și, dacă mă gândesc puțin, cred că va fi interesant de urmărit o carte literară creată de un colectiv de autori de text, sunet, grafică... O să ajungem oare să unificăm cartea cu filmul?

Îți mulțumesc și îți doresc succes.

(M.S.)



Multimedia. Enciclopediile pe CD merg însă dincolo de posibilitățile hârtiei și în ceea ce privește tipul de informații pe care-l oferă. Nu doar că, în general, numărul ilustrații lor – fotografii, desene, scheme etc. – este mult mai mare (pe hârtie, numărul și dimensiunile acestora sunt limitate de paginare), dar oferă și sunet, animații și chiar imagini video. Valoarea informativă a acestora este imensă. Puteți auzi ciripitul unei anume păsări exotice sau sunetul unui instrument muzical mai vechi, ca să nu mai vorbesc de fragmente din *Recviem*-ul lui Mozart sau din *Start Me Up* cu Rolling Stones. Puteți primi o lecție despre sistemul solar pe baza unei animații sugestive. Puteți vedea cursa de 100 de metri plat câștigată de Owens la Berlin în 1936 sau primii pași făcuți de Neil Armstrong pe Lună.

Hypertext. Este o practică uzuală ca articolele referite în cadrul altor articole să fie imediat accesibile, printr-un link. În acest fel, explorarea unui domeniu de interes devine mult mai comodă și mai eficientă decât în cazul trimiterilor din cărțile obișnuite.

Internetul

La momentul apariției primelor cărți pe CD-ROM, Internetul nu era încă o resursă foarte populară. Însă apariția și dezvoltarea extrem de rapidă a Web-ului a condus, destul de curând, la apariția primelor cărți în Internet. Ceea ce este demn de remarcat este că, deși dezavantajul major al cărții pe CD (necesitatea de a utiliza un calculator pentru lectură) se prelungește și în acest caz, Internetul este un mediu mult mai interactiv. Aceasta a permis unele experimente extrem de interesante, care evidențiază unele posibile mutații în industria cărții. În Internet este posibila revizuirea sau actualizarea imediată (sau permanentă) a textului publicat, spulberând astfel noțiunea clasică de ediție a unei opere. Mai mult, este posibil ca cititorii să participe la elaborarea unei „cărți" (vezi caseta „Opera deschisă").

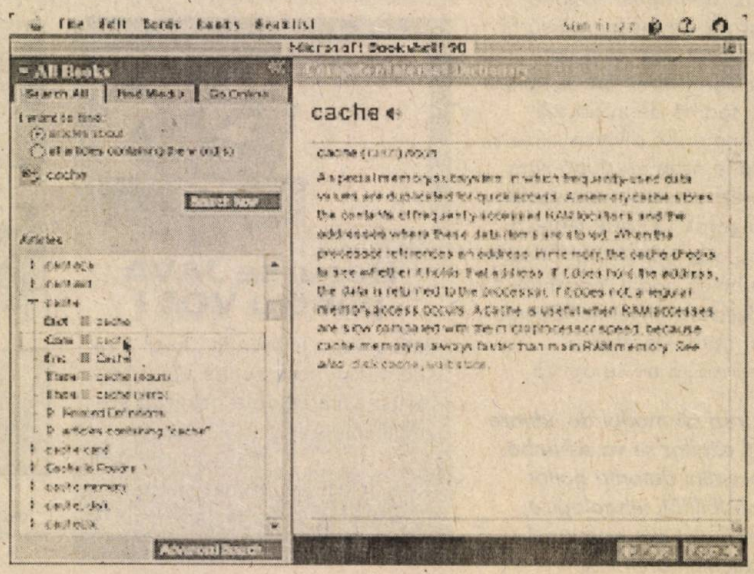
Transpunerea pe Web a unor cărți a devenit o practică destul de obiș-

Microsoft Bookshelf 98

Fără a fi atât de spectaculoasă și de interactivă ca Encarta, Bookshelf 98 este o bibliotecă de referință extrem de bogată și de utilă pentru oricine. "Raftul" (în engleză *bookshelf*) cuprinde nu mai puțin de 10 cărți de referință:

- *The American Heritage Dictionary of the English Language* (dicționarul de limbă);
- *Encarta 98 Desk Encyclopedia* (o ediție restânsă a enciclopediei din Encarta);
- *The Original Roget's Thesaurus of English Words and Phrases* (dicționar de sinonime);
- *The Columbia Dictionary of Quotations* (dicționar de citate);
- *Microsoft Bookshelf Computer and Internet Dictionary* (dicționarul explicativ de informatică editat de Microsoft Press);
- *The People's Chronology* (o colecție de biografii);
- *The World Almanac and Book of Facts 1997* (recapitularea celor mai importante evenimente ale anului 1997);
- *The Official Microsoft Bookshelf Internet Directory 98* (repertoarul de adrese Web);
- *Encarta 98 Desk Atlas* (o colecție de hărți);
- *National Five-Digit ZIP Code and Post Office Directory* (catalogul codurilor poștale din Statele Unite).

Criteriile de căutare oferite de Bookshelf se pot aplica asupra tuturor cărților sau asupra uneia anume. Numeroase legături la situri Web relevante pentru subiectul explorat sunt active atunci când dispuneți de o legatura la Internet.
O facilitate extrem de utilă este posibilitatea de a utiliza Bookshelf pentru căutare de referințe din cadrul altor programe, prin intermediul unui instrument specializat, Microsoft QuickShelf.
Ca și Encarta, Microsoft Bookshelf dispune de versiuni pentru Windows și Macintosh. Informații pe Web la adresa http://bookshelf.msn.com/.



nuită, în ciuda unor aspecte destul de delicate legate de drepturile de autor. Dicționarele și enciclopediile au devenit și ele prezențe comune pe Web (de pildă *Britannica Online* este disponibilă — contra unei taxe — pe Web, cu cele aproape 75.000 de articole, 7.000 de imagini, circa 23.000 de biografii, clipuri audio și video etc.).

Cu toate acestea, influența cea mai importantă și mai imediată pe care Internetul o exercită asupra lumii cărților este de altă natură. Ea se referă la modalitatea de a cumpăra cărți.

Amazonul

La mijlocul anului 1995 a luat ființă un sit Web care își propunea să fie o „librărie virtuală". Firma care-l deținea, Amazon.com, a dovedit perseverență și curaj (comerțul electronic era abia la începuturi) iar succesul a fost imens. Actualmente, catalogul Amazon.com (http://www.amazon.com) numără circa 3 milioane de titluri iar numărul clienților este uriaș.

Rețeta succesului a fost relativ simplă și s-a bazat pe două aspecte: posibilitățile de informare și prețul.

Căutarea unei cărți într-o librărie este adesea o experiență neplăcută. Zecile de mii de cărți sunt așezate după criterii destul de vagi iar căutarea după criterii ceva mai subtile decât poza de pe copertă este, de cele mai multe ori, o aventură. În schimb, cataloagele Amazon.com pot fi explorate după cele mai diverse criterii, fiecare carte este prezentată în detaliu începând cu poza copertei și terminând cu sumarul detaliat și eventual câteva pagini (prefața, introducerea etc.). Mai mult, sunt adesea disponibile textele recenziilor apărute în presă. Părerile cititorilor reprezintă o altă sursă importantă de informație. Ba chiar și autorul are dreptul să-și comenteze cartea. Pentru fiecare carte sunt recomandate și alte cărți pe același subiect sau având aceeași tematică.

În fine, o librărie virtuală nu are nevoie de un spațiu comercial de mari dimensiuni în centrul orașului (cu chiria de rigoare), nu are nevoie de un personal numeros și, în general, are cheltuieli mult mai reduse, ceea ce-i permită să vândă la prețuri mai mici decât librăriile clasice (la Amazon.com prețurile sunt cu 30%-40% mai scăzute decât în librării).

Comenzile se fac interactiv, pe situl Web, cărțile sunt expediate prin poșta de la cel mai apropiat depozit iar plata se face prin carte de credit.

Între timp au apărut și alte „fluvii" de acest gen, toate de succes, iar o parte dintre marile rețele de librării americane au adoptat și ele metoda. Ce se va întâmpla în aceste condiții cu librăriile „clasice"? Vor supraviețui doar cele care vor ști să se adapteze noilor condiții.

"Librăriile, desigur, nu mai sunt doar un loc de unde să cumperi cărți", observă Negroponte în coloana sa permanentă din Wired. „Ele sunt și pentru ca să frunzărești paginile cărților, ca să întâlnești oameni, să bei o cafea și să te lași în voia neprevăzutului". Este foarte posibil ca librăriile să se transforme din simple magazine într-un fel de „cluburi ale cărții", un loc de întâlnire pentru pasionați, un spațiu pentru manifestări culturale sau educative...

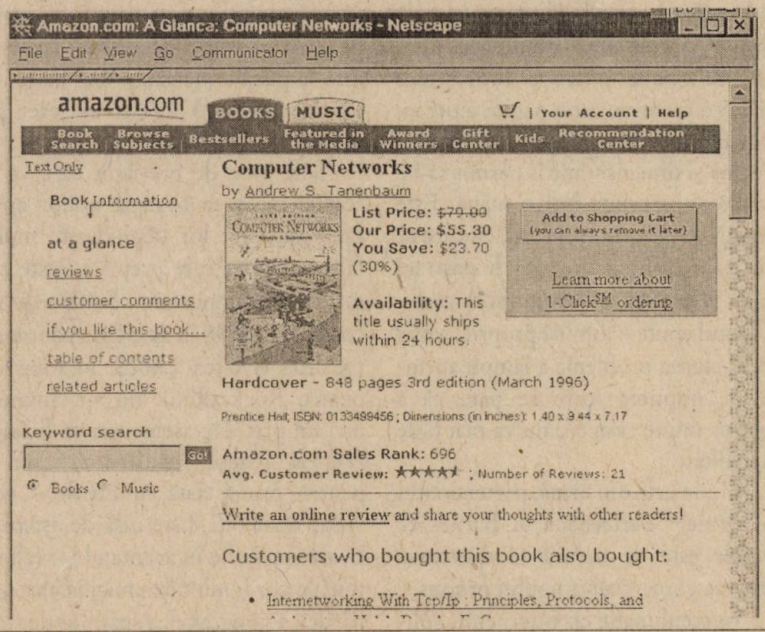
Cartea digitală

Se pare că, în ciuda vechimii, cartea ca obiect nu are încă rivali. Este ușoară, poate fi purtată cu noi în tren sau în avion, este ușor de manevrat și nu necesită un „User's Guide". Mai mult, în general este mai ușor de citit decât ecranul unui calculator și este suficient de ieftină. Și, poate cel mai important, este familiară.

Dintre obiecte, cărțile sunt cele de

Amazon.com

Librăria virtuală Amazon.com permite căutarea cărților după viverse criterii, cum ar fi numele autorului, titlul cărții, editura, subiectul, cuvinte cheie, etc. De asemenea, se pot consulta liste tematice pe diverse nivele de detaliere.
Odată găsită cartea, sunt oferite informații amănunțite despre formatul cărții (dimensiuni, modul de legare, număr de pagini), preț, disponibilitate, ediție etc. În general, se oferă cuprinsul detaliat al cărții și, eventual, pagini din prefață sau introducere. De asemenea, se pot consulta recenzii aparute în presă, comentarii ale cititorilor, se poate consulta o listă a cărților care au fost cumpărate de obicei împreuna cu cartea respectivă.



care oamenii se atașează cel mai adesea. „Ceva" din farmecul conținutului se răsfrânge într-un anume fel asupra obiectului. Marii cititori sunt adesea pasionați bibliofili, cărora textura hârtiei, saturația cernelii sau calitatea legăturii le spun multe. Caracterul personal al cărții este adesea subliniat și de însemnările și semnele pe care posesorul le-a făcut în timpul lecturii.

Și totuși, încep să apară alternativele. În această toamnă vor fi lansate pe piață trei variante de „aparate de citit" electronice. Toate au câteva caracteristici comune: sunt portabile, sunt ușor de folosit (au doar câteva butoane, în special pentru răsfoire înainte și înapoi), au memorie în care se pot încărca mai multe cărți.

Ideea de la care se pleacă se leagă tot de Internet. Utilizatorii acestor „cărți electronice" vor cumpăra cărți de la librării virtuale dar, spre deosebire de cumpărarea unei cărți clasice, în acest caz va fi livrat prin rețea doar conținutul cărții, care va fi stocat în memoria aparatului. Cartea poate fi apoi citită și, în cele din urmă, ștearsă din memorie (eventual, conținutul va putea fi stocat mai întâi într-o „bibliotecă virtuală" pentru a putea fi recitit mai târziu).

Proiectanții au aplicat toate trucurile pe care le-au învățat din eșecul unor tentative anterioare. Astfel, *RocketBook* (modelul dezvoltat de firma NuvoMedia) cântărește doar 560 de grame și are o formă care evocă o carte ale cărei pagini citite au fost întoarse peste coperta din spate. Forma și dimensiunile îi permit să fie ținută cu ușurință într-o mână. Este dotată cu un creion special, cu care se pot face notițe, care vor fi păstrate apoi împreună cu conținutul cărții. Ecranul emite o lumină proprie, astfel încât eterna problemă a lămpii aprinse pe noptieră (care se pare că a distrus multe căsnicii) nu va mai fi de actualitate.

Nu în cele din urmă, parteneriatul cu firmele Bertelsman și Barnes & Noble este de natură să sporească puterea comercială a noului produs.

În combinație cu Internetul, aplicațiile posibile sunt nelimitate. Iubitorii de lectură nu vor trebui să ia cu ei într-o călătorie o mulțime de cărți, ci doar să încarce conținutul volumelor dorite. Studenți vor putea încărca toate manualele pentru un semestru și poate că vor oferi sume mai mari pentru cărți deja adnotate de colegi mai mari sau chiar de profesori. Publicații efemere (cum sunt cataloagele de prețuri, manuale pentru software sau aparate etc.) vor putea duce o viață economică prosperă prin RocketBook și aparate similare. Ba chiar mai mult, autorii își imaginează cărți utilitare generate dinamic, la cererea... cititorului. De pildă un ghid turistic elaborat pe baza itinerarului, la care s-ar putea adăuga și un dicționar bilingv sau un ghid de conversație. Sau poate o carte de bucate specială, doar pe durata vizitei mamei-soacre...

În competiție vor intra încă două modele, cu soluții constructive mai mult sau mai puțin asemănătoare. *SoftBook*, produs de firma Virtual Press, este ceva mai mare (cântărește 1,3 kg.) dar este „legat" într-o copertă de piele (deschiderea acesteia pornește automat aparatul). Este de asemenea dotat cu un creion pentru adnotări și semne de carte. *Everybook* (proiectat de firma cu același nume) va avea două pagini (față în față, ca o carte deschisă) și își propune să reproducă nu doar textul, ci și grafica, cu maximă acuratețe. Aceasta însă implică dimensiuni mai mari (va cântări circa 1,5 kg) și un preț pe măsură.

Este greu de prevăzut care va fi reacția pieței la apariția noilor aparate. Succesul lor depinde de mulți factori, dintre care prețul nu este cel mai puțin important. SoftBook va costa de la 299$ în sus, Everybook se va situa la circa 1.600$, în timp ce pentru RocketBook nu s-a anunțat nici un preț. De asemenea, va fi interesantă reacția editorilor, care - pentru prima dată în istorie - vor vinde doar biți. Este însă de așteptat ca aceasta să fie în avantajul lor (cheltuielile vor fi mult mai mici în absența hârtiei, a tipografiei, a spațiilor de stocare etc.). Formatul în care va fi livrat conținutul este o altă factor încă nedecis. Everybook a anunțat că mizează pe PDF (un format de descriere a *imaginii* documentelor bazat pe PostScript), dar ceilalți competitori vor miza probabil pe formate derivate din SGML (un format dezvoltat de IBM pentru a descrie *structura* documentelor).

Avangarda

În laboratoarele Media Lab de la MIT (*Massachusetts Institute of Technology*) câțiva cercetători lucrează la ceea ce ei numesc „digital ink" (cerneală digitală), deși mai degrabă ar părea să fie hârtie digitală. În principiu, e vorba despre o folie asemănătoare hârtiei, impregnată cu milioane de particule, care poate să afișeze informații stocate în formă digitală. Sau, altfel spus, un display subțire cât o foaie de hârtie.

Câteva sute de astfel de file ar urma să formeze o carte electronică, care ar avea în cotor bateriile, memoria și conectorii pentru conectare la rețea. Principiul ar fi același ca și la cărțile descrise mai sus: se descarcă din Internet conținutul dorit, care va putea fi apoi „afișat" de filele cărții. Conținutul poate să fie orice: cărți, reviste, ziare sau doar simple rapoarte editate pe calculator (hârtia în sine devine... imprimantă!)

Nisip și siliciu

Într-o celebră povestire, numită *Cartea de nisip*, scriitorul argentinian Jorge Luis Borges vorbește de o carte al cărui număr de pagini era infinit. Nici una dintre paginile ei nu era prima și nici una nu era ultima. Numerotarea era absolut aleatorie și nici o pagină, odată văzută, nu mai putea fi regăsită.

Borges și-a trăit întreaga viață în bibliotecă, deși a orbit total în jurul vârstei de 50 de ani. Cartea de nisip reprezintă, poate, o metaforă a lumilor posibile prin puterea imaginației.

Cartea de siliciu va fi, la rândul ei, infinită. Toate cărțile omenirii vor putea fi cuprinse în ea. Biții se vor elibera de atomi, informația de materie.

Vă place GInfo?

Nume: _____ Prenume: _____ Vârsta: _____

Adresa: _____

Localitate: _____ Cod poștal: _____ Județ: _____ Telefon: _____

Ocupația: elev informatică ☐ elev ☐ student ☐ cadru didactic ☐ altă ocupație ☐ (precizați: _____)

Școala (firma, organizația etc.): _____

Dotări

Folosiți un calculator acasă? Da ☐ Nu ☐

Câte calculatoare sunt instalate în școală: (aprox.) _____

Aveți acces la e-mail: Da ☐ Nu ☐

Aveți acces la WWW: Da ☐ Nu ☐

Cum apreciați dotarea cu tehnică de calcul a școlii: _____

Aprecieri generale

Acordați note de la 1 la 10 pe următoarele criterii:

Structura tematică: ___

Calitatea articolelor: ___

Prezentare grafică: ___

Calitate tipografică: ___

Considerați că prețul este:

prea mare ☐ accesibil ☐ prea mic ☐

De cât timp cunoașteți Gazeta de Informatică (ani)? _____

Cum ați procurat acest număr:

chioșc ☐ abonament ☐ cadou ☐ altfel ☐

În acest număr

Apreciați cu note de la 1 la 10 articolele acestui număr:

Aritmetică cu numere mari (T. Socaciu) ___

Aritmetică cu numere mari (I. Cozac) ___

Despre cărți, școală și informatică ___

Memoria virtuală ___

Sunetul, o necesitate ___

Algoritmi geometrici elementari ___

Introducere în Linux ___

Internet și mass-media ___

Articolul care v-a plăcut cel mai mult: _____

De ce? _____

Articolul care v-a plăcut cel mai puțin: _____

De ce? _____

Considerați că ponderea problemelor rezolvate este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea problemelor propuse este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea subiectelor teoretice este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că nivelul de accesibilitate al articolelor este

prea cobotât ☐ potrivit ☐ prea ridicat ☐

În viitor

În următoarele numere așteptați:

mai multe articole de programare? Da ☐ Nu ☐

mai multe articole teoretice? Da ☐ Nu ☐

mai multe articole despre Internet? Da ☐ Nu ☐

mai multe prezentări de programe? Da ☐ Nu ☐

prezentări de jocuri? Da ☐ Nu ☐

articole despre hardware? Da ☐ Nu ☐

V-ar plăcea să citiți:

pagini redactate de elevii unor licee? Da ☐ Nu ☐

lucrări premiate, realizate de elevi? Da ☐ Nu ☐

actualități școlare? Da ☐ Nu ☐

prezentarea unor situri Web? Da ☐ Nu ☐

tips & tricks? Da ☐ Nu ☐

Comentarii:

_____

_____

_____

_____

_____

_____

_____

Dacă nu vă este suficient spațiul, folosiți coli albe suplimentare

Jocuri pentru toți

de Dragoș Marian

Prin amabilitatea firmei SC Multimedia Automex România SRL, distribuitor al editorului Cyberstone Entertainment (www.cyberstone.hu), am primit la redacție două CD-uri cu jocuri (ce bucurie pe noi!), apărute de curând într-o serie de CD-uri multimedia.

Primul lucru care ne-a plăcut la aceste CD-uri a fost interfața în limba română (exista și opțiunea pentru limba maghiară). Dispunând de un design interesant și prietenos, bine organizate, sunt foarte ușor de folosit. Jocurile sunt împărțite pe categorii: jocuri 3D, de acțiune, platformă, Shoot'Em Up, Gravity, Snake etc...

Acțiune

Primul CD este numit "Cele mai bune jocuri de acțiune". Denumirea este un pic exagerată, însă promite ore bune de joc. Am regăsit pe acest CD vechiul joc *Shoot'Em Up Raptor*, *Tyrian 2*, și mai vechile jocuri *Island Hooper*, *Future Dimension* etc. Jocurile 3D sunt reprezentate de *Anihilator Tank*, *Breakfree*, *HURL* (în stil Doom pentru copii, unde accentul este pus mai ales pe colectarea diferitelor obiecte, respectiv pe găsirea întrerupătoarelor sau a altor bonusuri), *Nanocore* (pentru Windows 95, un tenis de perete interesant). *Santana S. X-Mas 3D* este un joc cu o grafică excelentă (folosește Direct X) și de un comic aparte, în care personajul principal este un Moș Crăciun pornit pe împușcarea personajelor negative (spiriduși și "anti" Moș Gerilă). Pe acest CD se mai găsesc aproximativ 70 de jocuri interesante, nu foarte pasionante pentru un înrăit jucător, însă excelente pentru copii și pentru jucătorul începător sau mediu.

Perspicacitate

Al doilea CD primit este numit "Cele mai bune jocuri de perspicacitate". Vă asigur că acest CD vă va capta atenția ore întregi, relaxându-vă sau obligându-vă să vă concentrați. Categoriile acestui CD sunt cele obișnuite pentru jocurile de perspicăcitate (Boulder Dash, Gravity, Pac-Man, Snake, Tetris). Interesante, mai ales pentru copii, sunt "Jocurile cu broască", în care se cere multă îndemânare în manevrarea mouse-ului. Jocurile platformă sunt bine reprezentate de *Cyril Cyberpunk*, *Elmo* (în stil Nintendo Mario Bros), și *Fly* (joc cu o grafică excelentă, pentru Windows 95). Am descoperit câteva jocuri tetris excelente: *Diaggle Directions*, unde trebuie aranjate forme triunghiulare într-un spațiu triunghiular, *Tetris For Network 95*, care se joacă în rețea cu regulile normale ale tetris-ului, și *WordTris*, excelent pentru cei care cunosc limba engleză și învață franceză sau germană (trebuie scris cuvântul în franceză sau germană pornind de la cuvântul în limba engleză). CD-ul conține aproximativ un număr considerabil de jocuri (circa 90).

Și încă...

Ambele CD-uri conțin uneltele necesare pentru jocuri (drivere, DLL-uri): Direct X v.5.0, Win32, programe de arhivare (WinZip, WinRar), drivere de mouse și drivere pentru plăcile video S3. De asemenea, câțiva antiviruși (printre care și cunoscutul ThunderByte) pentru siguranță, deși versiunile pe care le veți găsi sunt destul de vechi.

vrei să fii pregătit pentru mâine?

FII CU OCHII PE NOI!

BYTE România • PCReport • CADReport • Ginfo

COMPUTER PRESS AGORA

Ne citesc SPECIALIȘTII. Actuali, în formare, viitori. ALĂTURĂ-TE ȘI TU!

HEWLETT® PACKARD

Seria de imprimante HP DeskJet 690 cu PhotoREt. Imaginea afacerilor rentabile.

Legenda spune că o imprimantă nu poate însufleți imaginile. Datorită unei strălucite combinații de tehnologii avansate, seria de imprimante HP DeskJet 690 spulberă odată pentru totdeauna această prejudecată. Imaginile imprimate au calitate fotografică datorită tehnologiei HP PhotoREt. Sistemul RealLife imaging asigură, cele mai vii culori şi cea mai bună calitate pentru reproducerile alb-negru. Mai mult decât atât! ColorSmart reglează automat nivelurile de culoare. Cu cartuşul Photo Cartrige, se obțin cele mai naturale culori, iar imaginile tale beneficiază de o calitate fotografică excepțională şi de o mai mare rezoluție. Nici n-apucă bine balaurul să pocnească din coadă şi documentele tale sunt deja tipărite, 5 pagini alb/negru sau 2 pagini color pe minut! Imprimanta HP DeskJet 690C reproduce imagini atât de reale de parcă simți arsura flăcărilor scoase de balaur. Aşa că, uită poveştile! Seria DeskJet 690 nu este una oarecare. Este sufletul documentelor tale!

IMPRIMANTELE HP. HÂRTIA ESTE ÎN SLUJBA TA.

DEALERI – Bucureşti: AMERILEX - 223 43 23 • MBL COMPUTERLAND - 336 75 99 • Q'NET - 211 78 55 • S&T ROMANIA - 330 73 20 • SYSTEM PLUS - 210 85 70 **THEMA** - 231 29 55 • **Bistrița:** LOGITEC COMPUTER - 063/23 27 96 • **Cluj:** SISTEC-CLUJ - 064/19 02 82 • **Timişoara:** BRIDGEMEN - 056/20 13 60
RESELLERI – Bucureşti: ALSYS DATA - 411 27 27 • BLUE RIDGE INTERNATIONAL - 210 68 28 • CIPRO - 220 04 00 • FLAMINGO COMPUTERS COM - 222 49 34 • FOCUS INTERNATIONAL - 322 30 69 NET CONSULTING - 638 33 45 • PRODUCTON - 312 30 78 • RADIX - 410 25 25 • SOFTWARE ITC - 232 44 71 • TRICORP ELECTRONICS - 679 48 79 • TOTAL SOFT - 230 09 76 **Baia Mare:** CUMULUS LOGIC - 062/43 07 35 • **Bacău:** MIDO IMPEX - 034/17 10 80 • **Braşov:** ICCO COMPUTERS - 068/14 43 54 • INTELPROF INTERNATIONAL - 068/41 13 54 SISTEM - 068/31 86 63 • **Cluj Napoca:** AT COMPUTER - 064/41 32 04 • **Craiova:** AS COMPUTERS - 051/16 32 80 • **Constanța:** FORTE DIGISOFT - 041/65 30 36 • GMB COMPUTERS - 041/61 92 22 **Deva:** ASCOPIER - 054/23 36 06 • **Galați:** ROMLOTUS - 036/47 40 00 • SPARK - 036/46 52 70 • **Hunedoara:** MEMORII TECH DATA SERVICE - 054/23 07 03 • **Iaşi:** RADIX - 032/ 21 24 53 • SERCOM - 032/12 25 89 **Piatra Neamț:** SOFT COMPREST - 033/ 22 21 21 • **Ploieşti:** PLATIN SISTEMS - 044/19 18 18 • **Reşița:** ELCOM - 055/22 07 11 • **Suceava:** PRONOVA SMART - 030/52 14 85 **Timişoara:** ETA–2U: 056/19 95 08 • NETRING COMPUTER - 056/13 28 94 • **Tg. Mureş:** ELECTRO ORIZONT - 065/16 65 72 • **Tulcea:** MICROLINE - 040/51 63 69
CENTRU DE SERVICE AUTORIZAT: 330 73 00

Gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

Toamna 1998
Vol. 8/6

Lei 8000

din sumarul numărului:

INDEXARE CU B - ARBORI

TEHNO
PROGRAME care se autoverifică

INTERNET
MOTOARE de căutare

SERIAL
LINUX pentru JAVA toți

PORTRET
Scurtă istorie a UNIX-ului

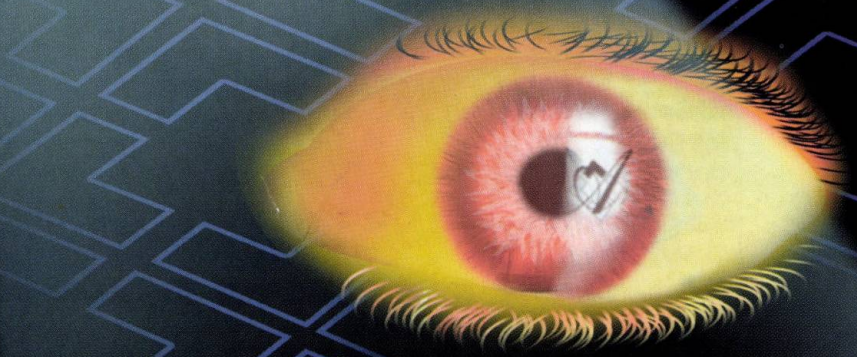
Probleme propuse și soluții **PLUS**
INTERVIU și altele

Expeditor
s.c."Computer Press Agora"s.r.l.
C.P. 172 O.P. 1
RO - 4300 Târgu-Mureș
T.P.
Aprobat R.A.P.R.
Nr.103/DC/053/1998
Valabil - 1998 -

INFADRES

vrei să fii pregătit pentru mâine?

FII CU OCHII PE NOI!



BYTE România • PCReport • CADReport • Ginfo

COMPUTER PRESS AGORA

Ne citesc SPECIALIȘTII. Actuali, în formare, viitori. ALĂTURĂ-TE ȘI TU!

Vol. 8, Nr. 6 1998

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu* (București)
Redactor șef:
prof. *Clara Ionescu* (Cluj)
Editor executiv:
Mircea Sârbu (Tg-Mureș)
Au mai colaborat:
Tiberiu Socaciu (Cluj)
Ioan Cozac (Târgu-Mureș)
Mihai Budiu (S.U.A)
Dan Iancu (București)
Cosmin Truța (Cluj-Napoca)
Cristian Mateescu (București)

Tehnoredactare:
Călin Borz
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
C.P. 172-1, 4300 Tg-Mureș
Adresa e-mail: office@ginfo.ro
Tel: (065) 166.516
Fax: (065) 166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *S.C. Infopress srl*,
Odorheiu Secuiesc,
tel: 066.216.483
Editor: Computer Press Agora srl
str. T. Vladimirescu nr. 63/1
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director editorial:
Mircea Sârbu
Director tehnic:
Szabó László
Vânzare reclamă:
Adrian Alexandru (aalex@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: (01) 330.92.81

Sumar

soluții.. PAG. 4

probleme REZOLVATE
PAGINI REALIZATE DE CLARA IONESCU
Probleme propuse la CEOI'98 și Balcaniadă

probleme.. PAG. 15

propuse pentru REZOLVARE
***
Lotul olimpic s-a antrenat la Constanța cu aceste probleme.

interviu.. PAG. 20

Noutăți în învățământ
Interviu cu d-ra *Anca Voicu.*

acrobați PAG. 22

Proiectul Netyear 2000
DE MIRCEA SÂRBU
Un proiect ambițios de informatizare a învățământului preuniversitar.

babel PAG. 54

Oberon: Magia componentelor
DE MIRCEA SÂRBU
Mai mult decât un limbaj: o filozofie.

focus.. PAG. 24

Indexare cu B-arbori
DE ION COZAC
O metodă de indexare care s-a impus prin performanță și versatilitate.

tehno.. PAG. 30

Programe autoverificabile
DE COSMIN TRUȚA

Despre Dangling-else
DE TIBERIU SOCACIU

portret.. PAG. 38

Unix - scurtă istorie
DE MIHAI BUDIU

serial.. PAG. 44

Excepții în Java
DE HORIA GEORGESCU

Linux: Procese
DE CRISTIAN MATEESCU

dialog.. PAG. 64

Chestionar

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.

Culegerea conține:

- probleme de algoritmică, majoritatea rezolvate în Pascal
- probleme de gestiune a bazelor de date (FOX-PRO)
- probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word
- probleme de verificare a cunoștințelor aferente sistemului de operare DOS

Această culegere constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru atestatul de informatică dar poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, culegerea conține o varietate mare de enunțuri de probleme rezolvate, abordând diferite metode și stiluri de programare.

• • •

Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) mai elegant și cu o eficiență în programare mai ridicată.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422] sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Înapoi la ȘCOALĂ

"Vacanțele sunt trecătoare, dar școala e eternă". Nu-mi mai amintesc cine a emis această sentință, dar în fiecare toamnă avem prilejul să-i verificăm valabilitatea.

În această toamnă însă am avut cu toții unele emoții. Ministerul Educației Naționale a început reforma învățământului, așa că multe lucruri nu mai sunt așa cum le-am lăsat în vară. Trimestrele s-au transformat în semestre, vacanțele s-au înmulțit și s-au mărunțit, bacalaureatul s-a înăsprit, iar despre examenele de admitere la facultate n-are rost să mai vorbim. Orele ar fi trebuit să se rărească, dar nu sunt semne că se va întâmpla așa ceva...

Primele rezultate ale reformei nu s-au lăsat așteptate și putem spune - fără riscul să greșim - că democrația a triumfat în școlile patriei: ceața e la fel de groasă de ambele părți ale catedrei. Și cum în vremuri tulburi zvonurile circulă mai repede decât știrile, ne-am gândit să le risipim cu câteva informații sigure, provenind din surse autorizate. ***Interviul*** din acest număr este dedicat în mare parte unor clarificări necesare privind învățământul de informatică.

Și cum informatică nu prea se poate face fără calculatoare, am încercat să aflăm cât mai multe date referitoare la proiectele de informatizare în domeniul învățământului. Proiectului "NetYear 2000" — menționat în numărul din vară al GInfo — îi este rezervată noua secțiune ***Actual***, dedicată actualităților din lumea școlară.

Secțiunea ***Portret*** s-a transformat pe nesimțite într-o rubrică dedicată istoriei informaticii, păstrându-și numele în virtutea tradiției și datorită faptului că întotdeauna istoria este dominată de personalități. Ce ar fi fost Unix-ul fără contribuțiile lui Ken Thomson sau ale lui Denns Ritchie?

Ca o confirmare a faptului că istoria este de fapt o știință a prezentului, am aflat că până și marii producători de sisteme de baze de date (Oracle, Informix și Sybase) au hotărât să-și dezvolte produsele și pentru Linux (un sistem de tip Unix) care câștigă tot mai mult teren, fiind actualmente cotat ca doilea sistem de operare ca răspândire în lume (după Windows). Așadar, ***serialul*** dedicat Linux-ului continuă.

La fel și serialul despre Java, care revine cu un episod dedicat unui lecții care se pare că nu prea se predă la clasă: siguranța executării programelor. Aceeași temă este abordată și în secțiunea ***Tehno***, prin intermediul articolul "Programe autoverificabile", aflăm despre aserțiuni, precondiții și postcondiții. Tot în această secțiune aflăm și despre pericolele care ne pândesc din spatele unei instrucțiuni aparent banale: IF.

De asemenea, siguranța execuției este unul dintre deziderateele principale ale limbajului prezentat în secțiunea ***Babel***. Oberon a fost adesea comparat cu Java, prelungind astfel mai vechea dispută privind superioritatea limbajului Pascal sau a limbajului C.

Secțiunea ***Internet*** are în acest număr un subiect foarte practic: cum să căutăm în imensitatea Rețelei informațiile care ne interesează. Motoarele de căutare reprezintă cu siguranță cea mai eficientă cale.

Și, bineînțeles, dacă a început școala au început temele pentru acasă. Secțiunea ***Probleme*** vă provoacă să vă încercați puterile cu câteva probleme cu care s-a antrenat și lotul olimpic. Secțiunea ***Soluții*** vă prezintă, ca de obicei, rezolvări ale unor probleme propuse în numerele trecute.

Am lăsat la urmă secțiunea ***Focus***, dedicată unui subiect care aparține în egală măsură algoritmicii și practicii curente a bazelor de date: indexarea cu B-arbori. Autorul articolului vă prezintă principiile, dar o implementare operațională și eficientă nu este ușor de realizat.

Poate fi o temă de trimestru. Pardon: de semestru! Încearcă cineva?

Mircea Sârbu

SOLUȚIILE problemelor propuse

În perioada 14-30 aprilie s-a desfășurat la Liceul de Informatică din Cluj tabăra de antrenament a lotului lărgit (informatică) având ca scop pregătirea și selecționarea echipelor participante la CEOI'98 și Balcaniada de Informatică. Vă prezentăm în continuare soluțiile problemelor date cu această ocazie.

P049801: Funcții surjective

Soluția autorului

(prof. Doina Rancea, Liceul de Informatică, Cluj)

Fie o funcție $f : N * N \rightarrow N$ definită prin $f(x,y) = b$, unde b este numărul funcțiilor surjective care pot fi definite de la o mulțime având cardinalul x, la o mulțime având cardinalul y.

Se demonstrează matematic foarte ușor că relația

$f(x,y) = f(x\text{-}1, y) * y + f(x\text{-}1, y\text{-}1) * y$

este adevărată pentru valori ale lui x mai mari decât 1. Observăm că pentru $x = 1$ avem $f(1,1) = 1$. Pe de altă parte se observă că pentru valori suficient de mari ale variabilelor x și y, valoarea b depășește 1020 și deci nu poate fi reprezentată intern, folosind tipurile standard oferite de limbajele de programare actuale, decât cu o anumită marjă de eroare. De aceea trebuie simulate operațiile de adunare a două numere mari și de înmulțire dintre un număr mare și un număr mic.

Se mai observă că dacă $y > x$ nu există funcții surjective de la o mulțime având cardinalul x la una ce are cardinalul y. Pornind de la formula recursivă pentru calcularea valorilor funcției f, putem ajunge, printr-un procedeu matematic destul de complicat, la termenul general al funcției f. Folosirea acestei formule ar implica implementarea mai multor tipuri de operații cu numere mari, ceea ce ar duce la scăderea performanțelor algoritmului.

Listing FUNCTII.PAS

```
Program FunctiiSurj;
#include <stdio.h>
#include <string.h>
class N {
private:
  int c[127],n; N prod(int);
  void transp(int); void init0();
public:
  N (unsigned long =0); N (char *);
  N& operator+=(N&); N& operator*=(N&);
  N& operator<<=(int);
  friend N operator+(N&,N&);
  friend N operator*(N&,N&);
  friend N operator<<(N&,int);
  friend void scrie(N&,FILE*);
};

N::N(unsigned long x){
  init0();
  for (n=0;x;n++) c[n]=x % 10, x/=10;
}
N::N(char* x){
  init0(); n=strlen(x);
  for (int i=0,nn=n-1;x[i];i++)
    c[nn-i]=x[i]-'0';
}
void N::init0(){
  for (int i=0;i<128;i++) c[i]=0;n=1;
}
N N::prod(int x){
  int i,t,s; N z=*this;
  for (i=t=0;i<n;i++){
    (s=x*z.c[i]+t)>9 ?
    (t=s/10,s %=10):(t=0);
    z.c[i]=s;
  }
  z.transp(t); return z;
}
void N::transp(int t){
  if(t) c[n]=t,n++;
}
N& N::operator+=(N& x){
```

```
  int i,t,s;if(n<x.n) n=x.n;
  for (i=t=0;i<n;i++){
    (s=c[i]+x.c[i]+t)>9 ?
    (t=1,s-=10):(t=0); c[i] = s;
  }
  transp(t);
  return *this;
}
N& N::operator*=(N& x){
  N s;
  for (int i=0;i<x.n;i++)
  if (x.c[i]) s+=(prod(x.c[i])<<i);
  return *this=s;
}
N& N::operator<<=(int x){
  if (x){
    for (int i=n-1;i>=0;i--) c[i+x]=c[i];
    for(i=0;i<x;i++) c[i]=0; n+= x;
      }
  return *this;
}
N  operator+ (N& x, N& y)
{ N z=x;z+=y; return z;}
N  operator* (N& x, N& y)
{ N z=x;z*=y;return z;}
N  operator<<(N& x,int y)
{ N z=x;z<<=y;return z;}
void scrie(N& x, FILE *f){
  for(int i = x.n - 1; i >= 0; i--)
    fprintf(f, "%d", x.c[i]);
  fputc('\n', f);
} void main()
{ N s[27],f[27],ss[10];
  s[1]=f[1]=1;
  int k,n,nr=0,i,j=0,nn[10],kk[10],
    necalc[10];
  char a[27],b[27],s0[]="0";
  FILE *f1,*f2;
  f1=fopen("in.txt","rt");
  f2=fopen("out.txt","wt");
  fscanf(f1,"%s%s",a,b);
  nn[nr]=strlen(a);
  kk[nr]=strlen(b);
  if (nn[nr]>=kk[nr]) necalc[nr]=1;
  else necalc[nr]=0;
  nr++; fclose(f1);
  for (n=2;n<27;n++) {
    s[n]=1;f[n]=n*f[n-1];
    for (k=n-1;k>1;k--)
      s[k]=s[k-1]+k*s[k];
    for (i=0;i<nr;i++)
      if (n == nn[i] && necalc[i])
      {ss[i]=s[kk[i]]*f[kk[i]];j++;}
    if (j == nr) break;
  }
  for(i=0;i<nr;i++)
    if (necalc[i]) scrie(ss[i],f2);
    else fprintf(f2,"%s\n",s0);
  fclose(f2);
}
```

Pentru a mări interesul pentru Java, vom prezenta o rezolvare în acest limbaj, propusă de Mihai Scorțaru, student, Universitatea Tehnică din Cluj:

```
import java.math.*;
import java.io.*;
class FunctiiSurjective{
  public static void main(String args[]){
    try{
      BufferedReader br=new BufferedReader
        (new InputStreamReader
        (new FileInputStream("in.txt")));
      String s=br.readLine();
      int x=s.length();
      s=br.readLine();
      int y=s.length();
      BigInteger[] a=new BigInteger[27];
      BigInteger[] b=new BigInteger[27];
      for (int i=0;i<27;i++){
        a[i]=new BigInteger("0");
        b[i]=new BigInteger("0");
      }
      a[1]=new BigInteger("1");
      for (int i=2;i<=x;i++){
        for (int j=1;j<=i;j++)
        b[i]=(a[i].multiply(new BigInteger
            (Integer.toString(j)))).add
          (a[i-1].multiply(new BigInteger
            (Integer.toString(j))));
        for (int j=1;j<=i;j++) a[i]=b[i];
      }
      PrintWriter pw=new PrintWriter
     (new FileOutputStream("out.txt"),true);
      pw.println(b[y]);
    } catch (IOException ioe)
      {System.out.println("IO Error");}
  }
}
```

P049802: Fractal

(Soluția autorului, student, Cătălin Frâncu, Universitatea Politehnică, București)

Problema este o generalizare a următoarei probleme: fie intervalul `[0,1)` care se împarte în trei părți egale. Partea din mijloc se elimină. Se repetă această operație la infinit, pentru fiecare din subintervalele rămase. Se cere să se determine dacă un număr rațional `p/q` aparține mulțimii rămase.

Dacă notăm cu A_i mulțimile obținute la pasul i, atunci avem:

$A_1=[0,1/3)U[2/3,1)$

...

și în plus: $A_\infty...A_n\subset...A_2\subset A_1$.

Pentru ca numărul rațional p/q să aparțină mulțimii A_1 trebuie ca prima cifră zecimală să fie pară. Pentru ca p/q să aparțină lui A_2 trebuie ca și a doua cifră să fie pară... Deci pentru ca p/q să aparțină lui A_∞ trebuie ca acesta (p/q) să aibă toate cifrele de după virgulă pare. Deoarece p/q este rațional înseamnă că partea lui zecimală va fi finită sau periodică, deci testarea apartenenței la mulțimea finală se poate face după un număr finit de pași. În plus, se observă că este necesar să testăm doar dacă primele q zecimale sunt pare.

Acum să ne ocupăm de problema generalizată în plan. Pentru a reduce numărul de calcule, simplificăm numerele (px, qx) prin cel mai mare divizor comun al lor și numerele (py, qy) tot prin cel mai mare divizor al lor. Deci numerele px, qx respectiv py, qy vor fi prime între ele. Acest lucru îl realizăm la citirea datelor cu ajutorul funcției cmmdc. Pătratele care se elimină sunt reținute în matricea Cut, astfel:

Cut[i,j]=true dacă pătratul (i,j) se elimină și Cut[i,j]=false în caz contrar. (i,j=1..3).

Aici nu testăm la fiecare pas dacă cifra zecimală corespunzătoare este pară, ci dacă perechea formată din (px*3 div qx, py*3 div qy) aparține unuia din pătratele care nu este eliminat. Numărul de pași (necesari pentru a vedea dacă avem sau nu soluție) este contorizat în max(qy,qx). Procedura TakeDigit va elimina la fiecare pas zecimala cea mai semnificativă din p/q, actualizând bineînțeles valoarea lui p.

Listing FRACTAL.PAS

```
Program Ratio;
type Matrix=array[0..2, 0..2] of Boolean;
var PX,QX,PY,QY,PXPeriod,PYPeriod,i:Integer;
    NIter:LongInt;
    Cut:Matrix;
    Ok:Boolean;

  Function Exp(Q:LongInt;K:Integer):Integer;
  { La ce putere e K in descompunerea lui Q }
  var IAux:Integer;
  begin
    IAux:=0;
    while Q mod K=0 do
    begin Inc(IAux); Q:=Q div K end;
    Exp:=IAux
  end;

  Function Cememedece(A, B:Integer):Integer;
  var IAux:Integer;
  begin
    while B<>0 do
    begin IAux:=A; A:=B; B:=IAux mod B end;
    Cememedece:=A
  end;

  Procedure ReadData;
  var C,NE,i,j:Integer;
  begin
    for i:=0 to 2 do
      for j:=0 to 2 do Cut[i,j]:=false;
    ReadLn(PX,QX); C:=Cememedece(PX,QX);
    PX:=PX div C; QX:=QX div C;
    ReadLn(PY,QY); C:=Cememedece(PY,QY);
    PY:=PY div C; QY:=QY div C;
    ReadLn(NE);
    for i:=1 to NE do
    begin
      Read(C);
      Cut[(C-1) mod 3,(C-1) div 3]:=true
    end;
    ReadLn
  end;

  Function Max(A,B:Integer):Integer;
  begin
    if A>B then Max:=A else Max:=B
  end;

  Procedure TakeDigit;
  var DX,DY:Integer;
  begin
    DX:=PX*3 div QX; PX:=PX*3 mod QX;
    DY:=PY*3 div QY; PY:=PY*3 mod QY;
    Ok:=Ok and not Cut[DX,DY];
    Inc(NIter)
  end;

Begin
  Assign(Input,'FRACTAL.IN');Reset(Input);
  while not SeekEof do
  begin
    ReadData;
    Ok:=True; NIter:=0;
    for i:=1 to Max(Exp(QX,3),Exp(QY,3)) do
      TakeDigit;
      PXPeriod:=PX; PYPeriod:=PY;
      repeat
        TakeDigit
      until ((PX=PXPeriod) and
        (PY=PYPeriod)) or not Ok;
      if Ok then Write('DA ')
            else Write('NU ');
      WriteLn(' -- ',NIter, ' iteratii')
  end;
  Close(Input)
End.
```

P049803: Cuvinte încrucișate

(soluție propusă de **Cătălin Frâncu**, *student, Universitatea Politehnică, București)*

Rezolvarea se bazează pe metoda *backtracking*, însă fără o optimizare corespunzătoare programul nu se va încadra în limita de timp impusă. De aceea, inițial vom extrage din re-

busul necompletat toate lungimile posibile ale cuvintelor care ar putea fi amplasate în el. Pentru aceasta ne vom folosi de o matrice (denumită d) cu 5 coloane și r>p linii, în care pe o linie vom reține informațiile despre o amplasare posibilă a unui cuvânt și anume:

- pe primele două poziții reținem poziția unde am putea amplasa prima literă a cuvântului
- pe a treia poziție reținem lungimea posibilă a unui cuvânt care urmează a fi amplasat
- pe a patra poziție avem 1 dacă acest cuvânt este plasat orizontal și 2 dacă este amplasat vertical
- pe a cincea poziție avem 1 dacă în căsuțele specificate de primele 4 poziții este amplasat un cuvânt și 0 în caz contrar.

Procedura care rezolvă efectiv problema se numește back și are ca prim parametru numărul de ordine al cuvântului care urmează a fi amplasat. Dacă această amplasare este posibilă, se trece la amplasarea următorului cuvânt (din cele date) : back(k+1).

Listing CAREU.PAS

```
{R-,I-,C-,S-}
Program Rebus;
uses Crt;
Type rebus=array[1..10,1..10] of char;
Var fin,fout:Text;
    i,j,k,n,m,p,pos:Word;
    b:array[0..11,0..11] of char;
    d:array[0..200,1..5] of Byte;
    c:array[1..5000] of String[10];
    a:rebus; f:Boolean;

  Procedure Afis(a:rebus);
  var i,j:Byte;
  begin
    for i:=1 to n do
    begin
      for j:=1 to n do Write(fout,a[i,j]);
      Writeln(fout)
    end
  end;

  Procedure back(k:Word;a:rebus);
  var i,j:Byte; b:rebus; f:Boolean;
  begin
    if k=m+1
    then
    begin afis(a); Close(fout); Halt end
    else
    begin
      for i:=1 to pos do
        if (d[i,5]=0) and
                    (d[i,3]=Length(c[k]))
        then
        begin
          if d[i,4]=1
          then
            begin
              f:=true; j:=0;
              while (j<d[i,3]) do
              begin
                if (a[d[i,1],d[i,2]+j]<>' ')
  and (a[d[i,1],d[i,2]+j]<> c[k,j+1])
                then
                begin f:=false; j:=d[i,3] end
                else j:=j+1
              end;
              if f
              then
              begin
                b:=a;
                for j:=1 to d[i,3] do
  b[d[i,1],d[i,2]+j-1]:=c[k,j];
                d[i,5]:=1;
                back(k+1,b);
                d[i,5]:=0
              end
            end;
            if d[i,4]=2
            then
            begin
              f:=true;
              j:=0;
              while (j<d[i,3]) do
              begin
                if (a[d[i,1]+j,d[i,2]]<>' ')
                   and (a[d[i,1]+j,d[i,2]]<>
                                c[k,j+1])
                then
                begin f:=false; j:=d[i,3] end
                else j:=j+1
              end;
              if f
              then
              begin
                b:=a;
                for j:=1 to d[i,3] do
                b[d[i,1]+j-1,d[i,2]]:=c[k,j];
                d[i,5]:=1;
                back(k+1,b);
                d[i,5]:=0
              end
            end
        end
    end
  end;

Begin
  Assign(fin,'CAREU.IN'); Reset(fin);
  Assign(fout,'CAREU.OUT'); ReWrite(fout);
  Readln(fin,n,p);
  FillChar(b,Sizeof(b),'-');
  FillChar(a,sizeof(a),' ');
```

```
  for i:=1 to p do
  begin
    Read(fin,k,j);
    b[k,j]:='@';
    a[k,j]:='@'
  end;
  for i:=0 to n+1 do
  begin
    b[0,i]:='@'; b[n+1,i]:='@';
    b[i,0]:='@'; b[i,n+1]:='@'
  end;
  pos:=0;
  for i:=1 to n do
    for j:=1 to n do
    begin
    if b[i,j]='@'
    then begin end
    else
    begin
      if b[i,j-1]='@'
      then
      begin
        k:=1;
        while b[i,j+k]='-' do Inc(k);
        pos:=pos+1;
        d[pos,1]:=i; d[pos,2]:=j;
        d[pos,3]:=k;d[pos,4]:=1;d[pos,5]:=0
      end;
      if b[i-1,j]='@'
      then
      begin
        k:=1;
        while b[i+k,j]='-' do Inc(k);
        Inc(pos);
        d[pos,1]:=i; d[pos,2]:=j;
       d[pos,3]:=k; d[pos,4]:=2; d[pos,5]:=0
      end
    end
  end;
  m:=0;
  while not SeekEof(fin) do
  begin Inc(m); Readln(fin,c[m]) end;
  for i:=1 to pos-1 do
  begin
    k:=i;
    for j:=i+1 to pos do
      if d[k,3]<d[j,3] then k:=j;
    d[0]:=d[k]; d[k]:=d[i]; d[i]:=d[0]
  end;
  back(1,a)
End.
```

P049804: Comerț interplanetar

(rezolvare propusă de
Mihai Pătrașcu, Colegiul Carol I, Craiova)

Pentru fiecare dintre planete se calculează numărul minim de planete diferite pe care o navă de transport ar trebui să le parcurgă pentru a ajunge pe Pământ. Valoarea exportului către Pământ se înmulțește cu 0.95 de fiecare dată când este necesară trecerea printr-una din planete. Se alege apoi planeta cu cel mai mare export către Pământ.

Listing COMERT.PAS

```
#include <stdio.h>
#include <mem.h>
struct planet_t {
  char p,f; double val; char a[26];
} a[26];
int read_data(FILE *f){
  int i,k,n; double x;
  char s[40],*p;
  if (fscanf(f,"%d",&n) != 1) return -1;
  memset(a,0,26 * sizeof(struct planet_t));
  for (i=0;i<n;i++) {
    fscanf(f,"%s %lf",s,&x);
    k=*s-'A';a[k].val=x;a[k].f=k;
    fgets(s,40,f);
    for(p=s;*p;p++) {
      if ((*p >= 'A') && (*p <= 'Z'))
        a[k].a[*p-'A']=1;
      else if (*p == '*') a[k].p=1;
    }
  }
  return 0;
}
void solve(void){
  int i,j,k,m;
  do {
    m=0;
    for (i=0;i<26;i++)
      for (j=0;j<26;j++)
        if (a[i].a[j] && (a[i].val*0.95-
            a[j].val > 0.00001)) {
          m=1;
          a[j].val=a[i].val*0.95;
          a[j].f=a[i].f;
          }
  } while(m);
}
void write_res(FILE *f){
  int i,from; double max=-1.0;
  for(i=0;i<26;i++)
    if (a[i].p)
      if(max<a[i].val)
        max=a[i].val,from=a[i].f;
  fprintf(f,"Import din %c\n";from+'A');
}
void main(void){
  FILE *f=fopen("IMPORT.IN","rt");
  FILE *fw=fopen("IMPORT.OUT","wt");
  while(!read_data(f)){
    solve();
```

```
    write_res(fw);
  }
    fclose(f);
    fclose(fw);
}
```

P049805: Aeroport

*(soluție propusă de **Mihai Scorțaru**, student, Universitatea Tehnică din Cluj)*

Până în prezent nu a fost descoperit un algoritm polinomial pentru rezolvarea acestei probleme, așa că o vom trata ca pe o problemă NP - completă. Algoritmul propus pentru rezolvarea acestei probleme va fi unul **probabilist**, mai precis un **algoritm genetic**.

Vom folosi o populație de zece cromozomi, cu o probabilitate de mutație de 0.2 ce se va mări la fiecare 1000 generații cu 10% și cu o probabilitate de recombinare de 0.6.

Structura unui cromozom va fi următoarea: va conține `nro` gene (`nro` reprezintă numărul de porți care vor fi folosite, adică numărul de orașe distincte care apar ca puncte de plecare sau ca destinații), fiecare element reprezentând indicele unei porți (un număr între 0 și $m * n - 1$).

Potrivit algoritmilor genetici pentru fiecare generație vom avea de parcurs trei etape: *selecția*, *mutația* și *recombinarea*.

Selecția se va realiza astfel:

- se calculează suma lungimilor drumurilor pentru configurația corespunzătoare fiecărui cromozom;
- se determină cromozomul cu cea mai defavorabilă configurație (fie `L` suma lungimilor drumurilor pentru acest cromozom);
- indicele de adaptare pentru fiecare cromozom va fi egal cu raportul dintre `L` și suma lungimilor drumurilor pentru cromozomul respectiv;
- se calculează suma `S` a indicilor de adaptare;
- fiecare cromozom va avea o probabilitate de a fi selectat egală cu raportul dintre indicele de adaptare al respectivului cromozom și valoarea `S`;
- împărțim intervalul [0,1) în zece subintervale astfel încât lungimea intervalului corespunzător unui cromozom să fie egală cu probabilitatea de selecție a acelui cromozom; de exemplu, pentru o populație care conține cinci cromozomi cu probabilitatea de selecție 0.3 și alți cinci cu probabilitatea de selecție 0.1, intervalele corespunzătoare vor fi: [0, 0.15), [0.15, 0.3), [0.3, 0.45), [0.45, 0.6), [0.6, 0.75), [0.75, 0.8), [0.8, 0.85), [0.85, 0.9), [0.9, 0.95), [0.95, 1);
- se generează zece numere aleatoare subunitare; pentru fiecare din aceste numere se determină intervalul din care face parte numărul, iar cromozomul corespunzător este adăugat la noua populație (se observă că un anumit cromozom poate fi ales de mai multe ori).

Pașii necesari simulării *mutațiilor* sunt următorii:

- pentru fiecare cromozom se generează un număr aleator subunitar; dacă acest număr este mai mic decât probabilitatea de mutație, atunci cromozomul este selectat pentru a deveni mutant;
- pentru un cromozom ales se selectează două gene și se modifică porțile corespunzătoare lor.

Pentru a realiza *recombinarea* vom parcurge următoarele etape:

- pentru fiecare cromozom se generează un număr aleator subunitar; dacă acest număr este mai mic decât probabilitatea de recombinare atunci cromozomul este selectat pentru a fi recombinat;
- se aleg perechi de câte doi cromozomi din lista celor selectați pentru recombinare;
- pentru fiecare pereche de cromozomi se alege un număr întreg `P` din intervalul `[1,nro-1]` care va indica faptul că fiecare cromozom va păstra primele sale `P` gene;
- informațiile corespunzătoare restului genelor sunt copiate din cromozomul pereche; în cazul în care apare situația ca o genă nouă să indice o poartă care există deja în structura unui cromozom, ea va fi înlocuită de cea mai apropiată poartă nefolosită;
- noii cromozomi își vor înlocui "părinții" în populație.

La fiecare generație vom păstra în populație cel mai bun cromozom obținut până atunci chiar dacă, în mod normal el ar fi fost pierdut.

Listing AERO.PAS

```
Program Aeroport_Algoritm_Genetic;
uses Crt;
const popsize=10;
      ppm=0.2;{Probab. initiala de mutatie  }
      pi=0.6; {Probabilitate de incrucisare }
type cromozom=array[1..100] of Byte;
var f:Text;
    pm:Real;
    a:array[1..100,1..100] of Byte;
    orase:array[1..100] of String[40];
    nrn,nri,nri1,m,n,nro:Byte;
    nr,gen,min,p:Longint;
    fol:array[0..100] of Boolean;
    p1,p2,sol,pr:cromozom;
   pp,pp1,pop:array[1..popsize] of cromozom;
    c:array[1..100] of Longint;
    c1:array[0..100] of Real;

Procedure afisare; forward;
  Function cod(var s:String):Byte;
  var i:Byte;
  begin
    for i:=1 to nro do
      if orase[i]=s
      then begin cod:=i; Exit end;
      Inc(nro); orase[nro]:=s; cod:=nro
  end;

Procedure citire;
var s1,s2:String; c:Char;
begin
  Assign(f,'AERO.IN'); Reset(f);
```

```
  Readln(f,n,m); nro:=0;
  while not SeekEof(f) do
  begin
    s1:='';
    repeat Read(f,c); s1:=s1+c until c=' ';
    s2:='';
    repeat Read(f,c); s2:=s2+c until c=' ';
    Readln(f,p); a[cod(s1),cod(s2)]:=p
  end;
  Close(f);
  if nro=0 then afisare
end;

Function cost:Longint;
var c:Longint; i,j:Byte;
begin
  c:=0;
  for i:=1 to nro do
    for j:=1 to nro do
      if a[i][j]>0
      then
        if pr[i] div m=pr[j] div m
       then Inc(c,a[i][j]*Abs(pr[i]-pr[j]))
        else Inc(c,a[i][j]*(pr[i] mod m +
             pr[j] mod m + 2 +
          Abs(pr[i] div m - pr[j] div m)));
  cost:=c
end;

Procedure afisare;
var i:Byte;
begin
  Assign(f,'aero.out'); Rewrite(f);
  Writeln(f,min);
  for i:=1 to nro do
    Writeln(f,orase[i],sol[i]+1);
  Close(f); Halt
end;

Procedure mutant(x:Byte);
var i,y,poz,poz1:Byte;
    a:set of Byte;
begin
  repeat
    poz:=Random(nro)+1; poz1:=Random(nro)+1;
  until poz<>poz1;
  a:=[];
  for i:=1 to nro do a:=a+[pop[x][i]];
  a:=a-[pop[x][poz]]; a:=a-[pop[x][poz1]];
  repeat
    y:=Random(m*n)
  until not (y in a ) and (y<>pop[x][poz]);
  pop[x][poz]:=y; a:=a+[y];
  repeat
    y:=Random(m*n)
  until not (y in a ) and (y<>pop[x][poz1]);
  pop[x][poz1]:=y; pr:=pop[x];
  if cost<min
  then
  begin min:=cost; Writeln(min); sol:=pr end
end;
Procedure mutatie;
var i:Byte;
begin
  for i:=1 to popsize do
    if Random<pm then mutant(i)
end;

Procedure imperecheaza;
var i,x,y,j,poz:Byte;
    a,b:set of Byte;
begin
  a:=[]; b:=[]; poz:=Random(nro-2)+1;
  for i:=1 to poz do
  begin a:=a+[p1[i]]; b:=b+[p2[i]] end;
  for i:=poz+1 to nro do
  begin
    x:=p1[i]; y:=p2[i]; j:=0;
    while true do
    begin
      if y+j < m*n
      then
        if not ((y+j) in a )
        then
          begin
            a:=a+[y+j]; p1[i]:=y+j; Break
          end;
      if y-j>=0
      then
        if not ((y-j) in a )
        then
          begin
            a:=a+[y-j]; p1[i]:=y-j; Break
          end;
      Inc(j)
    end;
    j:=0;
    while true do
    begin
      if x+j<m*n
      then
        if not ((x+j) in b )
        then
          begin
            b:=b+[x+j]; p2[i]:=x+j; Break
          end;
      if x-j>=0
      then
        if not ((x-j) in b )
        then
          begin
            b:=b+[x-j]; p2[i]:=x-j; Break
```

```
          end;
        Inc(j)
      end
    end
end;
Procedure recombina;
var i,x,a:Byte;
begin
  nri1:=nri; x:=nri div 2; i:=1;
  while i<=x do
  begin
    a:=Random(nri)+1; Dec(nri);
    p1:=pp[a]; pp[a]:=pp[nri+1];
    a:=Random(nri)+1; Dec(nri);
    p2:=pp[a]; pp[a]:=pp[nri+1];
    Inc(i);
    imperecheaza;
    pp[nri+1]:=p1; pp[nri+2]:=p2; pr:=p1;
    if cost<min
    then
      begin
        min:=cost; Writeln(min); sol:=pr
      end;
    pr:=p2;
    if cost<min
    then
      begin
        min:=cost; Writeln(min); sol:=pr
      end
  end
end;

Procedure Incrucisare;
var i:Byte;
begin
  nri:=0; nrn:=0;
  for i:=1 to popsize do
    if Random<pi
   then begin Inc(nri); pp[nri]:=pop[i] end
   else begin Inc(nrn);pp1[nrn]:=pop[i] end;
    recombina
end;

Procedure selectie;
var j,i:Byte; max:Longint; x:Real;
begin
  max:=0;
  for i:=1 to popsize do
  begin
    pr:=pop[i]; c[i]:=cost;
    if max < c[i] then max:=c[i]
  end;
  c1[0]:=0;
  for i:=1 to popsize do
    c1[i]:= c1[i-1]+1.0*max/c[i];
  for i:=1 to popsize do
    c1[i]:=c1[i] / c1[popsize];
  for i:=1 to popsize do
  begin
    x:=Random; j:=1;
    while c1[j]<x do Inc(j);
    pp[i]:=pop[j]
  end;
  pop:=pp
end;

Procedure generatie;
var i:Byte;
begin
  selectie;
  mutatie;
  Incrucisare;
  pop:=pp;
  for i:=nri1+1 to nro do
    pop[i]:=pp1[i-nri1+1];
  pop[popsize]:=sol
end;

Procedure crom(x:Byte);
var y,i:Byte;
begin
  FillChar(fol,Sizeof(fol),false);
  for i:=1 to nro do
  begin
    repeat y:=Random(m*n) until not fol[y];
    fol[y]:=true;
    pop[x][i]:=y
  end;
  pr:=pop[x];
  if cost<min
  then
    begin
      min:=cost; Writeln(min); sol:=pr
    end
end;

Procedure initializare;
var i:Byte;
begin
  for i:=1 to popsize do crom(i)
end;

Begin
  Randomize;
  citire;
  min:=Maxlongint;
  initializare;
  gen:=1;
  pm:=ppm;
  repeat
    if gen mod 1000 = 0 then pm:=pm*1.1;
    generatie;
```

```
    Inc(gen)
  until Keypressed;
  afisare
End.
```

P049806: Asemănare

soluție propusă de *Andrei Vancea*, *Liceul de Informatică*, Cluj

Două poligoane se numesc *asemenea* dacă au unghiurile congruente și laturile respectiv proporționale. În cazul particular al triunghiurilor se știe că este suficient să demonstrăm una din următoarele afirmații pentru ca două triunghiuri să fie asemenea:

- Triunghiurile au unghiurile congruente.
- Triunghiurile au laturile respectiv proporționale.
- Triunghiurile au câte două laturi respectiv proporționale și unghiurile formate de aceste laturi în aceste triunghiuri sunt congruente.

Un poligon cu N laturi poate fi triunghiularizat în N-2 triunghiuri. Dacă avem două poligoane pentru care cele N-2 triunghiuri sunt asemenea atunci putem afirma că cele două poligoane sunt asemenea. De aici deducem că primele două cazuri de asemănare pentru triunghiuri pot fi generalizate astfel:

- Două poligoane sunt asemenea dacă au unghiurile congruente.
- Două poligoane sunt asemenea dacă au laturile respectiv proporționale și diagonalele respectiv proporționale.

Programul prezentat aici folosește algoritmul bazat pe primul din aceste două cazuri de asemănare.

Listing POLIG.PAS

```
Program PoligoaneAsemenea;
type Punct=record
            x,y:Real
          end;
   MatriceDist=array[1..100,1..100] of Real;

var P:array[1..100] of Punct;
    Q:array[1..100] of Punct;
    UP:array[1..100] of Real;
    UQ:array[1..100] of Real;
    DP,DQ:^MatriceDist;
    N,i:Byte;
    csp:array[1..100] of Real;
    csq:array[1..100] of Real;

Procedure Citire;
var f:Text;
    i:Byte;
begin
  Assign(f,'POL.IN');
  Reset(f);
  Readln(f,n);
  for i:=1 to n do
    Readln(f,P[i].x,P[i].y);
  for i:=1 to n do
    Readln(f,Q[i].x,Q[i].y);
  Close(f)
end;
Function Dist(a,b:Punct):Real;
begin
  Dist:=Sqrt(Sqr(a.x-b.x)+Sqr(a.y-b.y))
end;

Procedure CalculeazaDistante;
var i,j:Byte;
    k:Byte;
begin
  FillChar(DP^,Sizeof(DP^),0);
  for i:=1 to n do
  begin
    k:=i+1;
    if k=n+1 then k:=1;
    DP^[i,k]:=Dist(P[i],P[k]);
    DP^[k,i]:=DP^[i,k];
    DQ^[i,k]:=Dist(Q[i],Q[k]);
    DQ^[k,i]:=DQ^[i,k];
    Inc(k);
    if k=n+1 then k:=1;
    DP^[i,k]:=Dist(P[i],P[k]);
    DP^[k,i]:=DP^[i,k];
    DQ^[i,k]:=Dist(Q[i],Q[k]);
    DQ^[k,i]:=DQ^[i,k]
  end
end;

Function Distanta(a:Punct;a1,a2:Punct):Real;
      { Distanta dintre punctul p fata de }
      { dreapta determinata de p1 si p2   }
var l1,l2,l3,p,s:Real;
begin
  l1:=Dist(a,a1);
  l2:=Dist(a,a2);
  l3:=Dist(a1,a2);
  p:=(l1+l2+l3)/2;
  s:=Sqrt(p*(p-l1)*(p-l2)*(p-l3));
  Distanta:=2*S/l3
end;

Procedure DetCos;
var i:Byte;
begin
  csp[1]:=(Sqr(DP^[2,n])-Sqr(DP^[1,n])-
       Sqr(DP^[1,2]))/2/DP^[1,2]/DP^[1,n];
  for i:=2 to n-1 do
    csp[i]:=(Sqr(DP^[i-1,i+1])-
         Sqr(DP^[i,i+1])-Sqr(DP^[i,i-1]))
             /2/DP^[i,i-1]/DP^[i,i+1];
    csp[n]:=(Sqr(DP^[1,n-1])-Sqr(DP^[1,n])-
            Sqr(DP^[n,n-1]))/2/DP^[n,n-1]/
            DP^[1,n];
```

```
    csq[1]:=(Sqr(DQ^[2,n])-Sqr(DQ^[1,n])-
             Sqr(DQ^[1,2]))/2/DQ^[1,2]/
             DQ^[1,n];

  for i:=2 to n-1 do
    csq[i]:=(Sqr(DQ^[i-1,i+1])-
           Sqr(DQ^[i,i+1])-Sqr(DQ^[i,i-1]))
            /2/DQ^[i,i-1]/DQ^[i,i+1];
    csq[n]:=(Sqr(DQ^[1,n-1])-Sqr(DQ^[1,n])-
             Sqr(DQ^[n,n-1]))/2/DQ^[n,n-1]/
              DQ^[1,n]
end;

Function Egal(a,b:Real):Boolean;
begin
  Egal:=Abs(a-b)<0.001
end;

Function OkUnghi(s:Byte; k:Integer):Boolean;
var i,x:Byte;
begin
  for i:=1 to n do
  begin
    if not Egal(csp[i],csq[s])
    then
      begin
        OkUnghi:=false;
        Exit
      end;
    Inc(s,k);
    if s=0 then s:=n;
    if s=n+1 then s:=1
  end;
  OkUnghi:=true
end;

Function OkLat(s:Byte; k:Integer):Boolean;
var prop:Real;
    x,i:Byte;
begin
  x:=s+k;
  if x=0 then x:=n;
  if x=n+1 then x:=1;
  prop:=DP^[1,2]/DQ^[s,x];
  for i:=2 to n-1 do
  begin
    s:=x;
    x:=x+k;
    if x=0 then x:=n;
    if x=n+1 then x:=1;
    if not Egal(DP^[i,i+1]/DQ^[s,x],prop)
    then
      begin
        OkLat:=false;
        Exit
      end
  end;
  OkLat:=true
end;

Procedure Scrie(s:Byte; k:Integer);
var f:Text;
begin
  Assign(f,'POL.OUT');
  Rewrite(f);
  for i:=1 to n do
  begin
    Write(f,s,' ');
    s:=s+k;
    if s=0
    then s:=n;
    if s=n+1
    then s:=1
  end;
  Close(f)
end;

Procedure ScrieNU;
var f:Text;
begin
  Assign(f,'POL.OUT');
  Rewrite(f);
  Writeln(f,0);
  Close(f)
end;

Begin
  New(DP);
  New(DQ);
  Citire;
  CalculeazaDistante;
  DetCos;
  for i:=1 to n do
  begin
    if OkUnghi(i,1) and OkLat(i,1)
    then
      begin
        Scrie(i,1);
        Halt
      end;
    if OkUnghi(i,-1) and OkLat(i,-1)
    then
      begin
        Scrie(i,-1);
        Halt
      end
  end;
  ScrieNU
End.
```

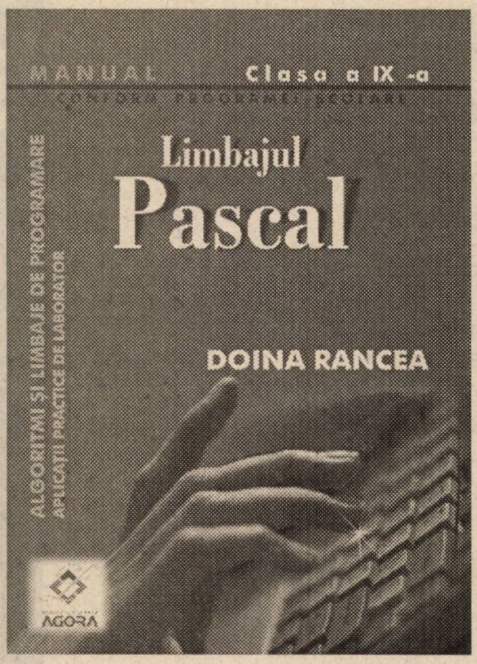
Paginile „Soluții" au fost coordonate de Clara Ionescu

E-mail: cionescu@ginfo.ro

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Limbajul Pascal este un manual care acoperă programa școlară (1997) a două discipline: „Algoritmi și limbaje de programare" și „Aplicații practice de laborator" care se studiază în clasele a IX-a cu profil de informatică.

Manualul prezintă noțiunile din programă la un nivel accesibil, fiecare noțiune fiind ilustrată în mai multe programe Borland Pascal 7.0. Aceste programe sunt însoțite și de prezentarea datelor de intrare și a rezultatelor în forma sub care apar pe ecran. Problemele au fost selectate în așa fel încât să ilustreze multiplele domenii de utilizare a calculatoarelor. Nu lipsesc problemele de simulare, jocuri, probleme distractive, dar au fost alese și aplicații din programa de matematică de clasa a IX-a cu scopul exemplificării și fixării unor noțiuni importante din matematică. În manual se insistă asupra felului în care se descoperă rezolvarea unei probleme, nivelul de dificultate a acestora crescând gradat, venind astfel în sprijinul elevilor și profesorilor.
Numeroasele probleme din manual permit utilizarea acestuia nu numai pentru studiu individual ci și ca bogat material didactic pentru lecțiile de programare și laborator.
Manualul se adresează elevilor, studenților, profesorilor și oricărei persoane interesate de programare.
Manualul a fost conceput, plecând de la ediția din 1993 (Limbajul Turbo Pascal, clasa a X-a) care a fost regândit substanțial și îmbogățit, astfel încât să fie accesibil unui elev de clasa a IX-a. Ediția anterioară s-a bucurat de un succes deosebit, fiind apreciată de mii de cititori.

Autoarea, Doina Rancea, este absolventă cu diplomă de merit a Facultății de Matematică, Secția Informatică a Universității Babeș-Bolyai din Cluj Napoca (1975). A fost repartizată la Liceul de Informatică din Cluj Napoca unde predă și în prezent limbajele de programare Pascal și C++. Este profesoară gradul I și în această calitate îndrumă activitatea de practică pedagogică a studenților. Experiența bogată a autoarei a fost valorificată cu ocazia participării, în cadrul Comisiei Naționale de Specialitate Informatică a Ministerului Educației Naționale, la actualizări ale programei școlare pentru disciplina informatică și în cadrul comisiilor județene și naționale ale olimpiadelor, propunând probleme originale.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

PROBLEME propuse pentru REZOLVARE

Între 26 iulie - 9 august, la Constanța s-a desfășurat a doua tabără de antrenament a lotului olimpic național de informatică. La sfârșitul primei săptămâni membrii lotului lărgit au primit șase probleme de concurs, la sfârșitul celei de a doua săptămâni au fost șapte probleme. Adresându-ne tuturor celor care iubesc algoritmica, vă propunem să vă încercați puterile! Rezolvările voastre se vor evalua și vă vom anunța punctajul obținut.

P069801: Măsurătoare

problemă propusă de prof. Eugen Ionescu, Liceul de Informatică, Cluj-Napoca

Fie două canistre având volumul `A`, respectiv `B` litri. Inițial amândouă canistrele sunt goale. Trebuie să punem `n` litri de apă în cea de a doua canistră efectuând următoarele operații permise:

`UA` se umple prima canistră de la robinet;
`UB` se umple a doua canistră de la robinet;
`GA` se golește prima canistră;
`GB` se golește cea de a doua canistră;
`AB` se toarnă apa din prima canistră în a doua, astfel încât dacă se poate turna toată cantitatea de apă, se toarnă tot, altfel atât cât încape, restul rămânând în `A`;
`BA` se toarnă apa din a doua canistră în prima, astfel încât dacă se poate turna toată cantitatea de apă, se toarnă tot, altfel atât cât încape, restul rămânând în `A`;

Să se determine (cunoscând volumele canistrelor și cantitatea de apă dorită `n` în cea de a doua canistră) secvența de operații care efectuate una după alta conduc la cantitatea cerută de apă în cea de a doua canistră. **Se cere secvența de operații de lungime minimă.**

Date de intrare:

În prima linie a fișierului de intrare `CANA.IN` este scris numărul `n` (cantitatea de apă dorită în canistra a doua) (`1≤N≤1000000`), iar în cea de a doua linie sunt scrise volumele canistrelor: două numere întregi `A` și `B`, despărțite printr-un spațiu (`1≤A,B≤1000000`).

Date de ieșire:

În fișierul de ieșire `CANA.OUT` se va scrie secvența de operații, o singură operație pe o linie. Dacă problema nu are soluție, atunci pe singura linie a fișierului se va scrie cuvântul `'NU'`. *Dacă soluția este corectă, dar secvența nu are lungime minimă, se acordă 20% din punctaj.*

Exemplu:

CANA.IN	CANA.OUT
2	UA
3 7	AB
	UA
	AB
	UA
	AB
	GB
	AB

Timp maxim de executare/test: 1 secundă

P069802: Nuclee

problemă propusă prof. Ioan Maxim, Liceul de Informatică, Cluj-Napoca

Fie `S` un șir de `n` numere naturale (`1≤n≤1000000`). Un nucleu de ordin `k` (`1≤k≤32000`) centrat în `m∈S` se definește ca fiind un subșir `K` al șirului `S` format din `k` elemente (nu neapărat consecutive) cele mai apropiate de `m` ($|x-m|$ este minim $\forall x \in K$).

Să se determine un nucleu al șirului `S` de ordin `k` centrat într-un element `m` din `S`.

Date de intrare

Fișierul de intrare `NUC.IN` conține pe prima linie numerele naturale `k`, reprezentând numărul de elemente ale subșirului `K` și valoare elementului `m` din șir, iar pe următoarele linii sunt scrise elementele șirului `S` (câte cel mult `50` de numere naturale pe o linie; valorile elementelor șirului sunt din intervalul `[1,65000]`).

Date de ieşire

În fişierul de ieşire NUC.OUT se vor scrie pe una sau mai multe linii elementele nucleului K.

Exemplu:

```
NUC.IN:
13 7
2 1 6 7 6 8 9 6 3 6 4 5 7 6 7 8 1 2 4 7 9
6 8 6

NUC.OUT:
6 7 6 8 6 7 6 7 8 7 6 8 6
```

Timp maxim de executare/test: 16 secunde

P069803: Electrotehnică

problemă propusă student Ovidiu Gheorghioiu, Universitatea Politehnică, Bucureşti

Unii dintre voi au probabil intenţia să meargă la facultate şi au ales deja o Universitate Politehnică. Una dintre principalele materii din primul an va fi *electrotehnica*. Printre multele lucruri interesante pe care le veţi învăţa în cadrul acestei discipline va fi *rezolvarea circuitelor*. Nu ar strica să aveţi un program care să vă ajute la această muncă, nu tocmai uşoară, nu? Să detaliem puţin.

Un *circuit* este alcătuit din *noduri* (numerotate de la 1 la *N*) şi *laturi* (numerotate de la 1 la *L*). Fiecare latură leagă două noduri (posibil egale!) într-un anumit sens (sensul este doar convenţional şi ajută la stabilirea semnelor). Pot exista mai multe laturi care leagă aceleaşi noduri. Pe laturi se află *elemente de circuit* de diverse tipuri. Vom considera doar două tipuri de elemente: *rezistoare* şi *surse de t.e.m.* (tensiune electromotoare). În funcţie de configuraţia circuitului, pe fiecare latură se stabileşte un curent. Rezolvarea circuitului constă în stabilirea intensităţilor acestor curenţi.

Dacă notăm cu *Ri* valoarea rezistorului de pe latura *i* (rezistoarele nu au un sens, deci $Ri \geq 0$), cu *Ei* valoarea t.e.m. (cu semn) şi cu *Ii* valoarea intensităţii curentului (de asemenea cu semn), cele 3*L* valori verifică *Legile lui Kirchhoff*:

Legea I: În fiecare nod, suma algebrică a intensităţilor adiacente este 0 (intensităţile se iau cu - (minus) dacă latura respectivă "intră" în nod, respectiv cu + (plus) dacă latura "iese" din nod).

Legea II: Pe fiecare buclă (ciclu) din circuit, suma algebrică a căderilor de tensiune pe rezistoare (cădere de tensiune = produsul *RiIi*) este egală cu suma algebrică a t.e.m.

Atenţie: o buclă poate conţine şi laturi luate în sens invers, caz în care intensitatea şi tensiunea respectivă se iau cu semn schimbat (nu şi valorile rezistoarelor!).

Firesc, este greu să aplicăm legile în forma de mai sus (mai ales *legea II* care poate conduce la *L*! ecuaţii). Este uşor de demonstrat că una (oricare!) dintre ecuaţiile *legii I* poate fi eliminată (cele *N* ecuaţii nu sunt liniar independente). De asemenea, în *legea II* este suficient să considerăm *L*-*N*+1 bucle independente (adică orice buclă să aibă o latură care nu apare în nici o altă buclă). Avem deci în total *L* ecuaţii cu *L* necunoscute.

În electrotehnică însă apar şi *sursele de t.e.m.* comandate. Acestea au valoarea variabilă, dependentă de alte valori din circuit. Vom considera doar surse care depind *liniar* de valorile unor intensităţi (sub forma *Ei* = *kIj*, *i*, *j* fiind laturi, iar *k* o valoare dată oarecare). Problema nu se complică însă prea mult, deoarece avem, pentru fiecare "extra" necunoscută, şi o "extra" ecuaţie.

Dându-se valorile (respectiv relaţiile) rezistoarelor şi t.e.m de pe fiecare latură, să se calculeze valorile intensităţilor de pe toate laturile.

Date de intrare:

Pe prima linie a fişierul ELTH.IN se dau N şi L ($1 \leq N \leq L \leq 100$). Pe fiecare dintre următoarele L linii se găseşte descrierea laturii respective: vârfurile pe care le leagă, urmate de valoarea rezistorului şi de două numere: dacă sursa este simplă, primul număr reprezintă valoarea ei, iar al doilea este 0. Dacă sursa este comandată, cele două numere corespund lui k şi j din relaţia Ei = kIj (i este numărul de ordine al laturii curente).

Date de ieşire:

Fişierul ELTH.OUT va conţine L linii. Pe fiecare linie i se va afla valoarea intensităţii curentului de pe latura i.

Observaţii:

Toate circuitele date sunt *conexe* şi *rezolvabile* (există soluţie unică);

Toate valorile (intrările şi ieşirile, exceptând bineînţeles valorile rezistoarelor) sunt numere *reale* în intervalul [-1000, 1000];

La ieşire se cer două zecimale exacte.

Exemplu:

```
ELTH.IN
2 3
1 2 1 5 0
1 2 1 2 3
2 1 1 0 0

Sistemul:
I1+I2-I3 = 0
R1I1-R2I2 = E1-E2
R2I2+R3I3 = E2
E2 = 2I3


ELTH.OUT
 0.00
```

```
5.00
5.00
-5.00
```

Timp maxim de executare/test: 2 secunde

P069804: Perechi stabile

problemă propusă prof. Emil Onea,
Liceul "Unirea", Focșani

La o petrecere sunt invitați N bărbați și N femei. Atât bărbații cât și femeile sunt numerotați de la 1 la N. Organizatorii doresc ca toți cei 2*N invitați să se simtă cât se poate de bine (să danseze cu o persoană preferată) și solicită din partea fiecărui invitat lista preferințelor sale. Această listă cuprinde numerele de ordine ale tuturor persoanelor de sex opus, ordonată descrescător în funcție de preferințe.

Numim *cuplare* lista formată din cele N perechi (bărbat, femeie) desemnate pentru dans. Spunem că o pereche arbitrară (*b*, *f*) dintr-o cuplare oarecare este ***stabilă*** dacă respectă condiția: orice femeie *f'* aflată înaintea lui *f* în lista preferințelor lui *b* și cuplată cu *b'* în cuplare, îl preferă pe *b'* lui *b*. Întreaga cuplare este ***stabilă*** dacă toate cele N perechi sunt stabile.

Exemplu:

Fie N=3 și următoarele liste ale preferințelor:

Barbat	Lista preferintelor
1	3 1 2
2	2 3 1
3	2 3 1

Femeie	Lista preferintelor
1	2 3 1
2	1 3 2
3	1 2 3

Fie în continuare următoarele două cuplări:

(1)

Barbat	Femeie
1	2
2	1
3	3

(2)

Barbat	Femeie
1	3
2	1
3	2

Cuplarea (1) nu este stabilă, deoarece perechea (1 2) nu este stabilă. Într-adevăr, bărbatul 1 preferă femeia 3 în fața partenerei sale (2), iar femeia 3 îl preferă pe bărbatul 1 înaintea partenerului ei (3). În schimb, cuplarea (2) este stabilă.

Cunoscând numărul N și listele de preferințe întocmite de cele 2*N persoane, **să se determine (dacă este posibil) o cuplare stabilă.**

Date de intrare:

Pe prima linie a fișierului PAIRS.IN este scris numărul natural N, N<=250.

Pe următoarele N linii se află listele de preferințe ale bărbaților sub forma unor permutări ale mulțimii {1,2,...,N}; numerele de pe linia i+1 reprezintă lista preferințelor bărbatului i în ordinea descrescătoare a preferințelor acestuia. Următoarea linie din fișier este vidă. Pe următoarele N linii sunt scrise preferințele femeilor în același format. Numerele scrise pe o linie sunt despărțite printr-un singur spațiu.

Date de ieșire:

Rezultatele se vor scrie în fișierul PAIRS.OUT pe N linii. O linie va conține o pereche de numere naturale din mulțimea {1,2,...,N} reprezentând un element din cuplarea stabilă determinată. Primul număr al fiecărei perechi este numărul de ordine al bărbatului iar cel de al doilea este al partenerei sale de dans. Dacă nu există soluție, în fișier se va scrie mesajul 'Imposibil'. În cazul în care există mai multe soluții, în fișier se va scrie una singură

Timp maxim de executare /test: 1 secundă

P069805: Wilhelm Tell

problemă propusă de studentul Cătălin Frâncu,
Universitatea Politehnică, București

Se dă un măr de forma unui poligon **convex** cu N ≤ 3000 laturi în planul *xOy*. În acest măr, Wilhelm Tell trage M ≤ 5000 săgeți. Săgețile descriu o traiectorie reprezentabilă printr-o dreaptă în planul *xOy*. Din păcate pentru el și pentru voi, Wilhelm Tell a îmbătrânit și nu mai vede chiar așa de bine, astfel că nu toate săgețile lovesc mărul. Să se spună câte săgeți își ating ținta și care este suma abaterilor pentru cele care nu își ating ținta.

Se consideră că o săgeată își atinge ținta dacă traiectoria ei are **cel puțin un punct comun** cu mărul. **Abaterea** unei săgeți de la țintă este egală cu distanța de la dreapta ei suport (Δ) la poligon (P): d(Δ,P)=min (d(A,B) symbol 124 \f "Symbol" \s 12
Asymbol 206 \f "Symbol" \s 12
symbol 68 \f "Symbol" \s 12
, Bsymbol 206 \f "Symbol" \s 12
P symbol 125 \f "Symbol" \s 12

Date de intrare

Fișierul de intrare TELL.IN conține următoarele date:

```
N
X1 Y1
...
XN YN
M
A1 B1 C1
...
AM BM CM
```

unde:

Xi și Yi sunt coordonatele **reale** ale vârfurilor poligonului care întruchipează mărul.

Atenție ! Punctele sunt date în ordine trigonometrică (în caz că vă servește la ceva). Poligonul este **strict** convex, în sensul că toate unghiurile sunt cuprinse în intervalul deschis $(0,\pi)$.

Ai, Bi și Ci sunt numere reale reprezentând parametrii dreptelor suport ale săgeților. Astfel, traiectoria săgeții cu numărul *i* este dreapta de ecuație

Date de ieșire

Fișierul de ieșire TELL.OUT va conține două numere. Primul din ele reprezintă numărul de săgeți care și-au atins ținta, iar al doilea reprezintă suma abaterilor (cu două zecimale exacte) pentru săgețile care nu și-au atins ținta. *Pentru fiecare din cele două numere se acordă 50% din punctaj.*

Exemplu:

TELL.IN:	TELL.OUT
5	3
2.0 4.0	3.41
-2 4	
-4 2	
-1 -2	
3 0	
5	
0 1 0	
1 -1 6	
2 1 -8	
1 0 -5	
1 -1 -5	

În acest exemplu, prima dreaptă "înțeapă" poligonul, a doua se află în prelungirea unei laturi a poligonului, a treia îl atinge într-un singur punct, iar a patra și a cincea trec la distanțe de 2 și respectiv $\sqrt{2}$.

Timp maxim de executare/ test: 2 secunde.

P069806: Călătorie cu bucluc

problemă propusă de prof. Roxana Tâmplaru, Liceul de Informatica, Craiova

Imaginați-vă că puteți călători în timp și puteți să vă transpuneți în "pielea" altor personaje. Presupuneți așadar că sunteți căpitanul unei corăbii care din păcate eșuează pe o insuliță izolată în mijlocul oceanului. Nici nu apucați să faceți câțiva pași și sunteți imediat înconjurat de băștinași care nu par nicidecum bucuroși de oaspeți. Sunteți apoi condus la căpetenia lor și deduceți, după o obositoare "discuție", că nu aveți o altă șansă de a scăpa cu viață decât să rezolvați o mică problemă sau un fel de joc de logică. Vă gândiți că totuși șansa vă surâde, doar vă considerați o minte ageră și nu credeți că niște "sălbatici" ar putea "inventa" ceva cu adevărat inteligent, care să vă pună în încurcătură. Oricum nu aveți altă șansă, așa că cereți mai multe amănunte referitoare la jocul respectiv. În fața dumneavoastră este adus un bătrân băștinaș care începe să vă "explice" ce aveți de făcut ca să vă salvați viața.

În cele din urmă, deduceți următoarele: în căsuțele unui caroiaj dreptunghiular sunt plasate grămăjoare formate din boabe de cafea (un bob de cafea nu poate aparține simultan mai multor căsuțe), iar alături de caroiaj se găsește o grămadă mare de boabe de cafea. Două căsuțe se consideră vecine dacă au o latură comună (asta înseamnă că orice căsuță din caroiaj poate avea cel mult patru vecini). Există două tipuri de operații pe care le puteți efectua asupra grămăjoarelor de boabe de cafea din căsuțele caroiajului:

- să adăugați un același număr de boabe (luate din grămada mare) la două grămăjoare plasate în căsuțe vecine;
- să luați același număr de boabe din două căsuțe vecine (boabele astfel extrase se pun în grămada mare), putând astfel chiar goli (elibera) anumite căsuțe din caroiaj.

De fapt, chiar acesta este scopul "jocului": folosindu-vă doar de cele două operații specificate mai sus, pornind de la o configurație impusă a caroiajului, să eliberați complet căsuțele de grămăjoarele de boabe de cafea plasate inițial în ele. Grămada de boabe plasată alături caroiajului este suficient de mare, astfel încât să permită în orice moment efectuarea unei operații de tipul 1).

Bătrânul vă "construiește" apoi o configurație a caroiajului și vă dă de înțeles că soluția trebuie furnizată până la apusul soarelui. Întrucât este foarte târziu, vă apucați imediat de treabă. Nimic mai potrivit în aceste momente decât laptop-ul dumneavoastră de care nu vă despărțiți niciodată. Practic, trebuie să decideți dacă există o succesiune de astfel de operații care să vă permită eliberarea tuturor căsuțelor, și dacă da, să generați o astfel de succesiune.

Date de intrare:

Fișierul IN.TXT conține datele de intrare:

- pe prima linie, separate printr-un spațiu, se găsesc dimensiunile caroiajului ($2 \le m, n \le 150$);
- pe următoarele m linii se găsește configurația caroiajului: fiecare astfel de linie conține n valori (numere naturale mai mici decât 1000), separate între ele printr-un singur spațiu, reprezentând, pentru fiecare linie a

caroiajului, numărul de boabe pentru fiecare căsuță în parte.

Date de ieșire

Rezultatul se va scrie în fișierul `OUT.TXT`, fiecare linie a sa fiind de forma: `L1,C1 L2,C2->Nb` unde:

- `L1` și `C1`, respectiv `L2` și `C2` reprezintă liniile și coloanele aferente căsuțelor vecine asupra cărora se efectuează operația specificată anterior prin numărul `Nb`. Valorile `L1`, respectiv `C1` sunt despărțite prin caracterul virgulă, valorile `C1` și `L2` printr-un singur spațiu, valorile `L2`, respectiv `C2` prin virgulă, iar valoarea `Nb` este precedată de caracterele `'->'`. Căsuța din colțul stânga sus este identificată prin `(1,1)`, căsuța din dreapta jos este identificată prin `(m,n)`, notația fiind cea uzuală de la lucrul cu matrice.

- `Nb` reprezintă numărul de boabe care se adaugă sau se extrage la / din două căsuțe vecine; `Nb` este număr întreg, pozitiv dacă se face o operație de tipul 1), respectiv negativ, în cazul unei operații de tipul 2);

Exemplu:

IN.TXT	OUT.TXT
3 3	1,1 2,1->-1
1 2 2	2,1 3,1->-2
4 3 2	2,1 2,2->-1
2 1 1	1,2 2,2->-2
	3,2 3,3->-1
	1,3 2,3->-2

Succesiunea operațiilor specificate în `OUT.TXT` are ca efect următoarele configurații intermediare (numerele "boldate" identifică celulele asupra cărora se efectuează operația indicată deasupra semnului "⇒"):

```
1 2 2  -1        0 2 2  -2
4 3 2  ⇒         3 3 2  ⇒
2 1 1            2 1 1

0 2 2  -1        0 2 2  -2
1 3 2  ⇒         0 2 2  ⇒
0 1 1            0 1 1

0 0 2  -1        0 0 2  -2        0 0 0
0 0 2  ⇒         0 0 2  ⇒         0 0 0
0 1 1            0 0 0            0 0 0
```

Observație:

În cazul unor configurații ce nu se pot elibera complet, în fișerul `OUT.TXT` se va scrie valoarea `0`.

Timp maxim de executare/test: 1 secundă

*Așteptăm soluții pe adresa redacției (**numai** prin e-mail), până la data de 1 noiembrie 1998.*

Noutăți în ÎNVĂȚĂMÂNT

Interviu realizat de Clara Ionescu

În această toamnă toată "suflarea" din școlile României trăiește efervescența noului și așteaptă cu nerăbdare să vadă concret cum se va aplica reforma anunțată de Ministerul Educației Naționale. Mass-media a popularizat modificările privind structura anului școlar, cele care implică sistemul de evaluare a elevilor și multe alte informații legate de această reformă. Câteva aspecte privind învățământul de informatică ne dezvăluie d-ra prof. ANCA VOICU, inspector de specialitate (informatică) la Ministerul Educației Naționale.

Clara I.: Acum, la început de an școlar, vă rugăm să ne spuneți în ce măsură se modifică învățământul de informatică și de când se vor aplica aceste modificări?

Anca Voicu: În propunerea privind planul cadru de învățământ pentru liceu apare *profilul matematică-informatică*. Noutatea constă în aceea că nu numai profilul s-a modificat, dar și modul în care va fi repartizat numărul de ore între discipline. De asemenea, proiectul planului de învățământ prevede un anumit număr de ore în trunchi comun, altele, opționale la alegerea școlii, respectiv a elevilor. Astfel, în trunchiul comun apar trei ore de informatică obligatorii și două ore de informatică în pregătirea opțională.

Cu siguranță, aceste schimbări au adus și nemulțumiri în rândul profesorilor de informatică, motiv pentru care voi face câteva completări justificative. Având în vedere faptul că societatea românească se îndreaptă spre o societate informatizată, sistemul educațional trebuie să ofere șanse egale tuturor celor care beneficiază de el. De aceea, noul plan de învățământ pentru învățământul primar și gimnazial oferă posibilitatea studiului disciplinei *"Tehnologia Informației"* ca disciplină opțională sau ca modul obligatoriu în cadrul disciplinei *"Educație Tehnologică"*. Astfel, la intrarea în liceu toți elevii vor avea un minim de cunoștințe privind utilizarea calculatorului. De aceea la profilul matematică-informatică, informatica trebuie privită ca știință, la fel ca și matematica. Noutatea pe care o vor aduce unitățile didactice din pregătirea opțională este aceea că ele vor putea avea caracter recuperativ, pentru egalitatea șanselor elevilor, de aprofundare a conținutului cuprins în pregătirea obligatorie sau de noutate din punct de vedere al conținutului. Totodată, este de subliniat și faptul că la toate profilurile învățământului liceal se regăsește disciplina *"Tehnologia Informației"*, care va avea caracter aplicativ, specific profilului liceului.

Clara I.: *Când se va aplica acest nou plan de învățământ?*

Anca Voicu: Ministerul Educației Naționale dorește introducerea acestui nou plan de învățământ începând din anul școlar 1999-2000.

Clara I.: *Este adevărat că se vor desființa liceele, respectiv clasele speciale de informatică?*

Anca Voicu: Nimeni nu a spus că se vor desființa liceele de informatică. Ele vor putea funcționa cu profilul matematică-informatică. De asemenea, liceele care au avut clase speciale de informatică vor putea să continue instruirea la nivel de profil.

Clara I.: *Credeți că introducerea disciplinei "Tehnologia Informației" este justificată la toate profilurile? Pun această întrebare deoarece se știe că o asemenea disciplină necesită echipament adecvat în școli. Vor fi dotate școlile în mod corespunzător pentru a face posibilă instruirea conform standardelor europene?*

Anca Voicu: Ministerul Educației Naționale are în acest moment o

strategie pentru informatizarea învățământului. Ea nu a putut fi aplicată în acest an datorită bugetului redus. Pentru anii următori se vor căuta și alte surse de finanțare pentru realizarea acestui deziderat.

Clara I.: *Este adevărat că va exista probă scrisă de bacalaureat din Informatică, sau se va adăuga bacalaureatului media obținută de absolvenți la disciplina informatică din clasa a XII-a?*

Anca Voicu: După cum se cunoaște, a apărut deja broșura orientativă pentru Bacalaureatul din anul 1999. Conform Legii Învățământului, se va susține proba de bacalaureat la disciplina de specialitate. Dacă se va hotărî altceva între timp, elevii vor fi anunțați în timp util.

Clara I.: *Ce ne puteți spune despre olimpiadele din acest nou an școlar? Se vor desfășura după o metodologie nouă? În caz afirmativ, în ce va consta schimbarea?*

Anca Voicu: Datorită schimbării structurii anului școlar, se va schimba calendarul de desfășurare a olimpiadelor școlare. Sunt posibile și modificări ale metodologiei de derulare a olimpiadelor. Sperăm să putem transmite în teritoriu toate schimbările până la data de 1 noiembrie 1998.

Clara I.: *Apropos de olimpiade: ani la rând lotul național de informatică a obținut rezultate deosebite la olimpiadele internaționale de informatică. Se știe că aceste olimpiade sunt, de fapt, de algoritmică și programare. Credeți că existența acestor olimpiade influențează pozitiv programele școlare dezvoltate și aplicate pentru disciplina informatică în țara noastră?*

Anca Voicu: Deoarece domeniul informaticii este foarte vast, au fost și vor mai fi multe dezbateri în ceea ce privește conținutul programelor școlare. Noul plan de învățământ permite rezolvarea acestei probleme în sensul că la profilul matematică-informatică se va păstra orientarea pe algoritmică și programare fără neglijarea părții de utilizare. La celelalte profiluri, orientarea va fi pe utilizare și, după caz, vor fi introduse elementele de algoritmică și programare.

Reîntorcându-mă la întrebare, într-adevăr tinerii olimpici informaticieni se întorc an de an cu medalii de la olimpiadele internaționale de informatică. Rezultatele lor sunt apreciate în țară și în străinătate, iar noi le dorim succes în continuare. Pe de altă parte aș menționa că un sistem de învățământ nu poate fi (generalizat) forțat la nivelul olimpicilor.

Clara I.: *Din cele relatate rezultă recunoașterea pe plan internațional a învățământului de informatică românesc. În acest context ce concursuri internaționale de informatică se vor organiza în viitor în România?*

Anca Voicu: În anul 2000, în România se va organiza Olimpiada de Informatică a Europei Centrale, concurs inițiat de România, la fel ca și Balcaniada de Informatică. La revenirea, în această toamnă, din Portugalia a lotului național de informatică (care ne-a reprezentat cu succes la Olimpiada Internațională de Informatică) ni s-a comunicat, că în cadrul ședinței Comitetului Olimpic a Olimpiadei Internaționale s-a propus, ca în anul 2007 România să găzduiască Olimpiada Internațională. Această propunere evident ne onorează, deoarece reprezintă recunoașterea la scară mondială a învățământului informatic din țara noastră.

Clara I.: Vă mulțumesc și vă doresc succes!

AncaVoicu: Vă mulțumesc și eu.

Carte de vizită

ANCA VOICU este absolventă a Universității Politehnice din București, secția Automatică și Calculatoare, anul 1992.

Este profesoară titulară la liceul "Grigore Moisil" din București și, din 1995, este inspector de specialitate (informatică) la Ministerul Educației Naționale.

Aveți întrebări?

Dacă interviurile noastre v-au trezit interesul și v-au stârnit curiozitatea, cu siguranță v-ați gândit la întrebări pe care noi nu le-am pus. Încă nu e târziu. Trimiteți-le prin e-mail la redacție (adresa: ginfo@agora.ro) iar noi le vom trimite destinatarilor și vom colecta răspunsurile. Pe cele mai interesante le vom publica.

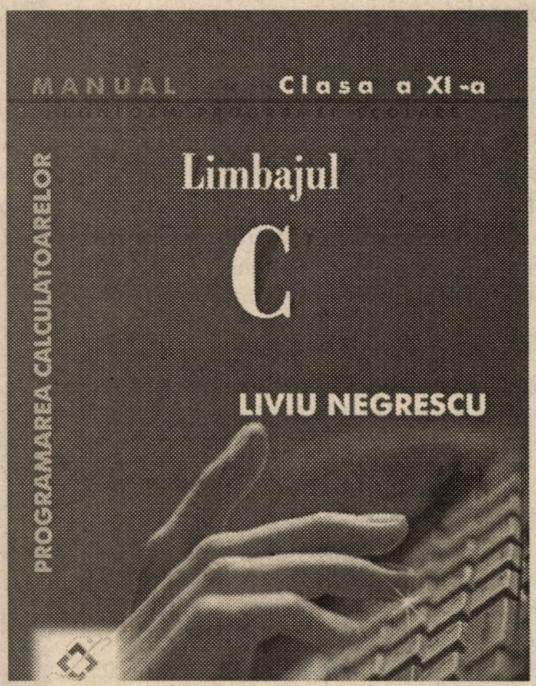
Așadar, are cineva întrebări pentru Monica Ene-Pietroșanu, Irina Athanasiu sau Anca Voicu? Le așteptăm.

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie*

... un MANUAL.



Această carte este concepută în conformitate cu Programa analitică a clasei a XI-a, profilul Informatică. Ea poate fi considerată o introducere elementară în limbajul C. Fiecare noțiune este explicată în mod clar și utilizată în programe simple. De asemenea, capitolele sunt însoțite de exerciții propuse care contribuie la fixarea și înțelegerea noțiunilor întâlnite în capitolele respective.

Cartea se adresează atât elevilor cât și celor interesați să se inițieze în limbajul C care în prezent este unul dintre cele mai utilizate limbaje de programare.

Liviu Negrescu este cercetător principal gr. I la societatea SOFTWARE ITC S.A., filiala Cluj, unde se ocupă cu dezvoltări de programe de sistem și aplicative.
În același timp, desfășoară o activitate didactică prestigioasă de peste 15 ani, la Facultatea de Calculatoare de la Universitatea Tehnică din Cluj.
În decursul anilor a fost titular a mai multe cursuri, ca de exemplu:

- Limbaje de programare și compilatoare;
- Proiectarea translatoarelor etc.

Liviu Negrescu are o serie de lucrări științifice publicate în diferite reviste de specialitate. În colaborare cu alți cercetători de la SOFTWARE ITC S.A. a implementat sisteme cu aplicații în diferite domenii:

- Construcția de compilatoare;
- Interfețe utilizator;
- Prelucrări de date din domenii economice și medicale.

Liviu Negrescu este autorul mai multor cărți apărute în decursul anilor, cu privire la diferite limbaje de programare (CPL1-subset al limbajului PL/1, CDL-limbaj de descriere al compilatoarelor precum și referitoare la limbajele C și C++. Lucrarea beneficiază de experiența vastă a autorului câștigată atât în activitatea de cercetare cât și în cea didactică.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Proiectul NetYear 2000

Pornind de la modelul unei acțiuni similare, desfășurate în Marea Britanie, firma ICL a demarat un proiect amplu de informatizare a școlilor din România.

Desfășurat sub deviza "*Connecting our children to a better future*", proiectul UK NetYear a fost caracterizat de premierul Tony Blair ca fiind cel mai mare parteneriat public și privat pentru realizarea unui sistem educațional, din toată lumea și din toate timpurile. Inițiat de firmele ICL, Sun Microsystems, Cisco Systems și de ziarul The Daily Telegraph, proiectul UK NetYear 2000 urmărește informatizarea și conectarea la Internet a tuturor școlilor din Marea Britanie, astfel încât în anul 2002 Internetul să poată fi larg folosit în educație. Pe lîngă autoritățile britanice, proiectul a fost susținut de numeroși sponsori. Spre exemplu, BBC Education este implicat într-un amplu proces de instruire pentru profesori (desfășurat sub deviza "Calculatoarele nu mușcă") iar Excite UK oferă oricărui elev un cont de e-mail gratuit, sprijinind astfel unul dintre obiectivele proiectului ("până în anul 2002, 75% dintre profesori și 50% dintre elevi să aibă o adresă de e-mail").

Aflat în plină desfășurare în Marea Britanie, proiectul a fost adaptat și inițiat în România sub denumirea "NetYear 2000" de către firma ICL, prin intermediul reprezentanței sale în România. Obiectivul proiectului constă din dotarea a circa 16.000 de școli cu câte o rețea formată dintr-un server și 15 stații de lucru, la care se adaugă conectarea la Internet.

Compania a obținut acordul Guvernului și al Ministerului Educației Naționale, a obținut deja circa 40% din finanțare și se află la ora actuală în stadiul de "organizare de șantier". Încă nu este sigur dacă se va forma sau nu o organizație specializată pentru punerea în practică a proiectului, deocamdată funcționează un Consiliu de Coordonare Națională la care participă reprezentanți ai Ministerului Educației, ai Comisiei Centrală de Informatică, ai finanțatorilor și, desigur, ai firmei ICL.

Proiectul se va întinde pe o durată de 6 ani și va costa circa 300 de milioane de dolari. Obiectivul pentru anul 1998 este informatizarea unei școli "prototip", urmând ca apoi să înceapă "producția de serie", cu o cadență de 50-60 de școli pe lună. În paralel va fi pusă la punct și infrastructura din teritoriu. Este interesant de semnalat că, pentru școlile din zone în care nu sunt acoperite de rețeaua telefonică (!), se prevăd soluții alternative (fie prin GSM, fie prin viitoarea rețea de telefonie rurală).

Interesant este de remarcat că, de fapt, partea de instalare a echipamentelor este partea cea mai ușoară. Proiectul nu se referă doar la aceasta, ci prevede încă două componente extrem de importante: instruirea profesorilor și dotarea cu software. Absența acestora ar compromite tocmai obiectivul final al întregului proiect: accesul profesorilor și a elevilor la o tehnologie educațională modernă și la resursele informaționale uriașe ale Internetului, în scopul creșterii calitative a procesului de învățământ. Tehnologia Informației este privită ca un instrument universal, care poate fi utilizat la orice nivel și formă de învățământ, la oricare materie.

În privința instruirii, este prevăzută o schemă piramidală. În prima etapă va fi instruită în Marea Britanie o echipă de "instructori de instructori" care, ulterior, va forma instructori care, la rândul lor, vor urma să transmită profesorilor informațiile tehnice de care aceștia vor avea nevoie pentru utilizarea efectivă a facilităților pe care le vor avea la dispoziție. ICL caută la ora actuală și oferte complementare în domeniul instruirii, deoarece consideră că există pericolul ca acest proces să se desfășoare mai lent decât dotarea efectivă a școlilor.

În privința soft-urilor, dotarea va include sistemul de operare (gama Windows) și utilitarele obișnuite. Pe lângă acestea, se prevede dotarea cu o gamă largă de aplicații cu caracter educațional. Datorită particularităților firești ale învățământului din România (privind programa școlară, tehnologiile didactice etc.) o mare parte a acestor softuri educaționale nu pot fi preluate sau adaptate din afară. Proiectul prevede și sume importante care să fie folosite pentru crearea de software educațional în țară, fapt ce ar putea trezi interesul firmelor românești de software către acest domeniu, care s-ar putea dovedi profitabil.

Cu speranța că învățământul românesc are, totuși, șansa de a ține pasul cu lumea civilizată, merită să amintim în final cuvintele lui Tony Blaire: *Children cannot be effective in tomorrow's world if they are trained in yesterday's skills.*

(M.S.)

Indexare cu B-arbori

De Ion Cozac

Accesul rapid la date este o preocupare majoră a programatorilor de aplicații de baze de date și a administratorilor de baze de date, cu atât mai mult cu cât volumul datelor care trebuie prelucrate este tot mai mare. Acest articol prezintă fundamentele unor structuri de date proiectate în acest scop.

De ce folosim indecși?

Indecșii sunt o soluție pentru un acces mai rapid la date, tot așa cum cuprinsul unei cărți sau reviste ne ajută să găsim mai repede informația care ne interesează, fără să fim obligați să răsfoim întreaga carte sau revistă. Acești indecși sunt importanți în special în cazul SGBD-urilor relaționale (Paradox, Fox, Access, Oracle ș.a.), deoarece în lipsa acestora este necesară parcurgerea unui procent important din tabelă pentru localizarea informațiilor dorite, și acest proces este de cele mai multe ori ineficient. Parcurgerea cu ajutorul indecșilor este mult mai eficientă deoarece, în general, o intrare în tabela index ocupă mult mai puțin spațiu decât înregistrarea corespondentă din tabela de date. În concluzie, utilizarea indecșilor asigură o reducere considerabilă a operațiilor de intrare/ieșire necesare localizării informațiilor dorite. Acest aspect este cu atât mai important cu cât accesul la disc este foarte lent (de ordinul milisecundelor)în comparație cu accesul la memoria centrală (de ordinul nanosecundelor).

Pe de altă parte, gestionarea indecșilor cere o atenție deosebită. Cu toate că indecșii sunt foarte utili pentru regăsirea datelor, aceștia pot diminua performanțele unui SGBD (Sistem de Gestiune a Bazelor de Date) în cazul operațiilor de scriere. Aceasta deoarece o modificare a unui câmp folosit pe post de cheie necesită două operații de scriere: una pentru a reflecta modificarea în tabela de date și alta pentru a reflecta modificarea în tabela index. În concluzie, dacă operațiile asociate unei baze de date sunt în mare parte scrieri, este importantă o utilizare atentă a indecșilor pentru tabelele relevante (astfel de cazuri apar mai ales atunci când se implementează o nouă aplicație de baze de date și este necesară încărcarea tabelelor principale). Indecșii necesită de asemenea un volum de spațiu pe disc, fapt care trebuie avut în vedere la alocarea resurselor pentru baze de date.

Ce este un B-arbore?

Un B-arbore (în engleza *B-tree*) este o structură de date care întreține un set ordonat dedate (chei) și garantează operații eficiente pentru căutarea, inserarea, ștergerea și parcurgerea datelor. În cadrul acestui articol, fiecare informație înregistrată într-un B-arbore va fi numită ***cheie***, deoarece fiecare cheie este unică și poate apărea în B-arbore într-un singur loc. Într-o implementare practică informația înregistrată într-un B-arbore se compune din valoarea cheii și un pointer către înregistrarea corespondentă din fișierul de date propriu-zise, și eventual lungimea înregistrării (în cazul în care fișierul de date se compune din înregistrări de lungime variabilă).

Un B-arbore este o structură multinivel care conține un nod rădăcină ale cărui intrări indică spre nivelul imediat inferior al nodurilor din B-arbore. Cel mai de jos nivel păstrează nodurile frunză (vezi figura "Exemplu de B-arbore").

Proprietățile unui B-arbore de ordin n:

- fiecare nod conține între n și 2nchei (astfel nodurile sunt cel puțin pe jumătate ocupate), cu excepția nodului rădăcină care poate conține mai puțin de n chei;
- fiecare nod conține între n+1 și 2n+1 pointeri către nodurile fii (aflate pe nivelul imediat inferior),

cu excepția nodurilor frunză care nu au asociate noduri fii;

• cheile apar în ordine crescătoare în interiorul fiecărui nod;

• o listă de pointeri este dispersată printre chei pentru a indica unde să se caute o cheie atunci când aceasta nu se află în nodul curent.

B-arborii sunt o generalizare a arborilor binari. Într-un B-arbore fiecare nod care nu e frunză are cel puțin două noduri fii, plasate după următoarele reguli:

• două noduri fii se află în stânga și în dreapta unei chei al nodului părinte;

• nodul fiu din stânga conține cheile cu valori mai mici decât cheia nodului părinte, și cel din dreapta conține cheile cu valori mai mari decât aceasta.

Un aspect important al B-arborilor este faptul că distanța dintre nodul rădăcină și fiecare nod frunză este aceeași și această proprietate se păstrează indiferent de numărul de inserări și ștergeri precum și de numărul frunzelor. Această uniformitate previne dezechilibrarea nodurilor și nivelelor arborelui, astfel că arborele rămâne balansat (acum ne dăm seama de unde provine termenul de B-arbore). Parcurgerea arborelui pentru a regăsi o anumită informație va dura cam la fel de mult indiferent unde este localizată aceasta.

Dacă un B-arbore rămâne tot timpul balansat, nu același lucru se întâmplă și în cazul arborilor binari. Căutarea într-un arbore binar nebalansat (în care distanța dintre nodul rădăcină și nodurile frunză variază între niște limite destul de largi) poate necesita traversarea unui număr mare de noduri. De exemplu, arborele poate avea o structură aproape liniară dacă înregistrările se memorează aproximativ în ordinea cheilor.

Căutarea unei chei într-un B-arbore începe întotdeauna cu nodul rădăcină și urmează pointeri din nod în nod până când fie cheia este localizată, fie căutarea eșuează atunci când se ajunge la un nod frunză și nu mai avem pointeri de urmat.

Inserarea unei chei

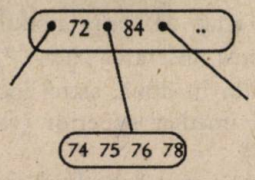
Să considerăm B-arborele prezentat ca exemplu. Pentru a insera cheia "59", începem prin a o căuta. Dacă ea este găsită, inserarea nu mai are rost (nu e cazul nostru). Dacă nu, am ajuns la un nod frunză în care ar trebui să se afle această cheie. În cazul nostru inserăm pur și simplu cheia la locul ei, astfel că nodul frunză arată astfel: ("58", "59", "60", "61").

În urma acestei operații nodul devine plin, dar acest fapt nu ne deranjează.

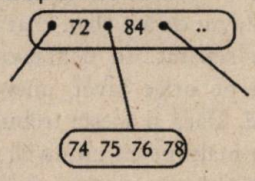
Să inserăm acum cheia "77". Căutarea inițială ne

Inserarea unei chei

Înainte de inserare:



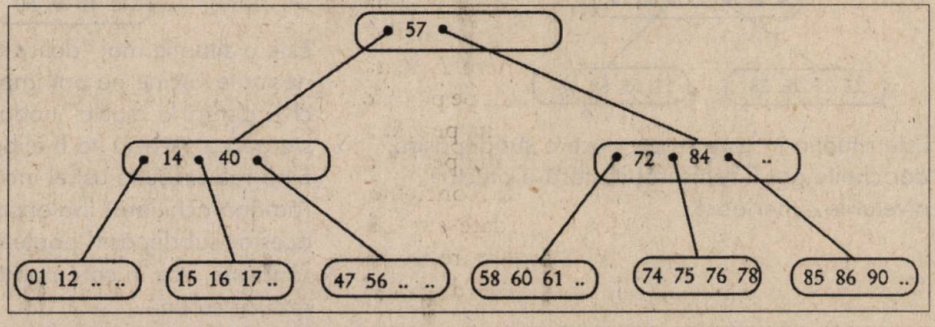
După inserare:



Inserarea cheii "77" produce separarea nodului, deoarece nu a existat suficient spațiu pentru adăugarea simplă a cheii în cadrul nodului. La modul practic aceste cazuri sunt foarte puține în comparație cu situațiile simple. Există însă și situații mai "ghinioniste", când separarea se propagă pe nivelurile superioare, uneori până la nodul rădăcină.

Exemplu de B-arbore

Exemple practice de B-arbori sunt foarte greu de exprimat pe hârtie, datorită dimensiunilor foarte mari (număr de chei înnod, număr de noduri, număr de nivele etc.). Cel prezentat are o valoare strict didactică, fiind utilizat doar pentru a ilustra conceptele introduse în articol.



conduce la nodul frunză unde cheia ar trebui inserată, dar nodul este deja plin, având 4 chei: ("74", "75", "76", "78"). Adăugând încă o cheie s-ar încălca regula conform căreia ordinul B-arborelui este 2 și nu poate avea mai mult de 4 chei. Datorită acestei condiții de depășire (a limitei superioare - *overflow*), nodul frunză trebuie separat în două noduri. Primele două chei din stânga înlocuiesc vechiul nod, ultimele două din dreapta formează un nou nod frunză, și cheia din mijloc este "promovată" pe nivelul imediat superior, fiind inserată în nodul părinte aflat deasupra nodului frunză curent. În cazul nostru, inserarea cheii "77" conduce la separarea nodului în două, astfel încât cheia "76"trece pe nivelul imediat superior (vezi figura "Inserarea unei chei").

În acest caz, nodul părinte a avut doar două chei, fiind suficient spațiu pentru a "promova" și insera cheia "76". Dacă acesta ar fi avut deja 4 chei, el ar fi trebuit de asemenea să fie separat. În concluzie, separarea se poate propaga pe orice nivel, uneori chiar până la nodul rădăcină. Dacă și acesta trebuie separat, B-arborele crește în înălțime cu un nivel, și se formează un nou nod rădăcină cu o singură cheie.

Ștergerea unei chei

Un B-arbore se micșorează (se restrânge) atunci când se șterg chei. Pentru a șterge o cheie, mai întâi se efectuează o operație de căutare pentru a localiza nodul care o conține (în cazul în care cheia nu este găsită, ștergerea nu mai are rost).

Dacă cheia căutată se află într-un nod care nu este frunză, se caută cheia următoare (conform relației de ordine definite peste mulțimea valorilor cheilor), și ele își vor schimba locurile. Într-un B-arbore, cheia următoare este întotdeauna prima cheie din cel mai din stânga nod frunză al sub arborelui din dreapta. Să privim din nou B-arborele dat ca exemplu: următoarea cheie după "57" este "58", la care se ajunge urmând pointerul din dreapta cheii "57", după care mergem tot timpul spre stânga, urmărind cel mai din stânga pointer în fiecare nod până când întâlnim nodul frunză. Interschimbând cele două chei, putem muta cheia pe care dorim să o ștergem într-un nod frunză, după care operația de ștergere se simplifică. În felul acesta nu se afectează ordinea cheilor dată de pointeri în întregul B-arbore.

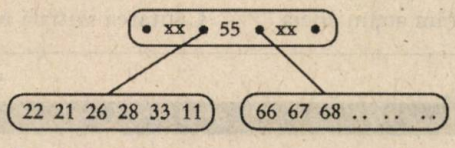
Odată ce cheia căutată se află într-o frunză, aceasta poate fi ștearsă. Dacă în nod rămân cel puțin n chei, totul este în regulă, în caz contrar se produce o subdepășire (a limitei inferioare - *underflow*).

Dacă într-un nod se produce o subdepășire, putem redistribui cheile împrumutând câteva chei de la nodul vecin. De exemplu, în B-arborele (care are ordinul 3) din figura "ștergerea unui nod" (cazul a.), dorim să ștergem cheia "67", producând astfel o subdepășire. În acest caz, cheile din nodul vecin stâng se deplasează prin nodul părinte și sunt redistribuite astfel încât fiecare nod frunză va avea în final 4 chei:

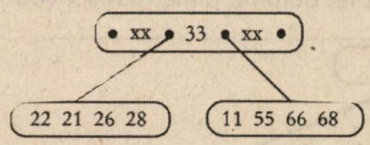
Dar dacă nodul curent și cel vecin au împreună mai puțin de 2n chei pentru redistribuire, cele două noduri se vor combina și vor forma unul singur. De exemplu, în B-arborele de ordin 2 din figura "ștergerea unui nod" (cazul b.) dorim să ștergem cheia "52". Acest fapt produce o subdepășire în

Ștergerea unei chei

Cazul a.
Înainte de ștergere:

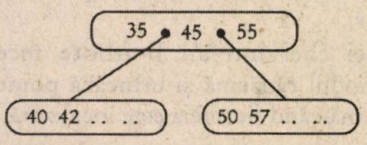


După ștergere:

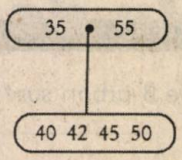


Este situația în care se produce o subdepășire, dar cheile pot fi redistribuite fără a afecta nivelurile superioare.

Cazul b.
Înainte de ștergere:



După ștergere:



Este o situație mai "delicată", deoarece nodurile vecine nu pot împrumuta suficiente chei pentru a "salva" nodul din care se face ștergerea. Acesta va fi eliberat, cheile rămase fiind redistribuite astfel încât arborele să rămână echilibrat (balansat). Propagarea acestor subdepășiri poate ajunge la nivelul rădăcinii, caz în care numărul de niveluri se reduce cu unu.

nodul curent, și nodul vecin nu poate împrumuta chei pentru redistribuire. În acest caz un nod este eliberat, și o cheie din nodul părinte coboară un nivel pentru a umple singurul nod rămas:

În acest caz, prin scoaterea cheii "45" din nodul părinte rămân doar două chei ("35" și "45"). Dar dacă nodul părinte are din start doar n chei, atunci și în acest caz se produce subdepășire când cheia se deplasează în jos pentru a intra în componența nodului imediat inferior. Observăm că subdepășirea și combinarea nodurilor se poate propaga până la nodul rădăcină. Dacă și nodul rădăcină produce subdepășire (acesta rămânând gol), atunci înălțimea arborelui (numărul de nivele) scade cu unu iar nodurile aflate pe al doilea nivel se combină pentru a forma o nouă rădăcină.

Observație. Dacă nodul frunză curent este cel mai din stânga dintre nodurile fii, în caz de necesitate se împrumută chei de la vecinul din dreapta. Trebuie să existe un vecin în dreapta deoarece orice nod care nu este frunză are cel puțin două noduri fii. Ca o strategie generală, în cazul ștergerii unei chei dintr-un nod frunză care nu se află pe margine, se va alege cea mai bună soluție atunci când se impune un "împrumut". Cea mai bună soluție este aceea care conduce la un număr minim de operații (cel puțin la nivelul nodurilor frunză). Evident, în cazul nodurilor aflate pe margine nu există alternative.

Statistici

Să considerăm un B-arbore de ordin n care se desfășoară pe k nivele. Un nod poate avea între n și 2n chei, deci între n+1 și 2n+1 pointeri (pentru analiza pe care o vom face mai jos ignorăm situațiile de excepție). O primă concluzie: un B-arbore este ocupat în proporție de 75% în medie: între 50% (toate nodurile ocupate pe jumătate) și 100% (toate nodurile complet ocupate).

Care este numărul de chei care poate fi memorat (și numărul de înregistrări corespondente care pot fi referite)?

Studiem mai întâi cazul când nodurile sunt ocupate pe jumătate. Pe nivelul 0 (nodul rădăcină) avem un nod, n chei și n+1 pointeri. Pe nivelul 1 avem n+1 noduri (cele referite de pointerii aflați pe nivelul 0), n(n+1) chei și $(n+1)^2$ pointeri. Deducem că pe nivelul k (al nodurilor frunză) avem $(n+1)^k$ noduri și $n(n+1)^k$ chei. Numărul total de înregistrări care pot fi referite este de

$$n\cdot[1+(n+1)+...+(n+1)^{k+1}]=$$

$$n\cdot\frac{(n+1)^{k+1}-1}{n+1-1}=(n+1)^{k+1}-1$$

În cazul în care toate nodurile sunt complet ocupate, numărul total de înregistrări este de $(2n+1)^{k+1}-1$.

Putem deduce ușor că numărul M de înregistrări care pot fi referite verifică relația:

$(n+1)^{k+1}-1 \le M \le (2n+1)^{k+1}-1$

Exemplu: dacă n=50 și k=3, atunci numărul de înregistrări este între 6250000 și 100000000.

Pentru un fișier cu M înregistrări care folosește un B-arbore de ordin n, numărul de nivele al B-arborelui este de $\log^n M$.

Pentru a identifica o înregistrare având o cheie dată, numărul maxim de accese la disc este de cel mult k+1 pentru parcurgerea nivelelor arborelui, și eventual încă unul pentru citirea înregistrării propriu-zise (în caz de succes). Scanarea (parcurgere secvențială) unui nod necesită cel mult $\log^2 n$ accese la memoria centrală (care este mult mai rapidă decât discul); am presupus că în interiorul unui nod se face o căutare binară, fapt posibil datorită ordonării crescătoare a cheilor în cadrul acestuia. Dacă presupunem că accesul la memoria centrală durează o unitate de timp, și accesul la disc durează D unități de timp (D este de ordinul sutelor de mii), atunci o înregistrare este localizată în $(k+1)(D+\log^2 n)+D$ unități de timp. Ne putem ușor da seama că valoarea D are un ordin de mărime mult mai mare decât $\log^2 n$. Așadar putem presupune că o înregistrare poate fi localizată cu succes în cel mult (k+2)D unități de timp. Dacă înregistrarea căutată nu există, răspunsul vine tot după (k+2)D unități de timp.

Alte operații

Până acum am descris câțiva algoritmi strict necesari pentru gestionarea unui B-arbore: localizarea, inserarea, ștergerea unei chei. Această structură poate fi completată pentru a putea efectua ușor și alte operații, cum este de exemplu parcurgerea înregistrărilor în ordine crescătoare (sau descrescătoare), eventual cu limitarea domeniului valorilor cheilor căutate.

Fiecare nod (cu excepția nodului rădăcină) se completează cu un pointer spre nodul părinte; această structură astfel obținută o întâlnim și la arborii binari și ea se numește arbore "însăilat". Putem parcurge în post-ordine un B-arbore însăilat, astfel:

- mai întâi toate cheile aflate pe nodul frunză cel mai din stânga;
- prima cheie din nodul părinte;
- toate cheile aflate pe al doilea nod frunză (pe care îl găsim urmând pointerul aflat după prima cheie din nodul părinte);
- a doua cheie din nodul părinte ș.a.m.d.;
- de fiecare dată când epuizăm un nod, urcăm pe nivelul imediat superior și selectăm următoarea cheie a acestuia, după care trecem la cel mai din

stânga nod frunză al acestuia;

• procedeul se repetă până când epuizăm ultimul nod frunză al arborelui.

Această tehnică de parcurgere ne dă ordinea crescătoare a cheilor din B-arbore.

Aplicație. Uneori, datorită numeroaselor operații de inserare și ștergere, B-arborele ajunge să fie destul de "rarefiat", mai ales dacă nu folosim o tehnică de recuperare a spațiului eliberat (*garbage collection*). Se poate proiecta un algoritm care reorganizează și compactează B-arborele, astfel:

• structura noului B-arbore va fi identică cu cea veche: va avea aceeași organizare pe nivele, fiecare nod va conține aceleași chei ca și corespondentul său vechi; singura deosebire este că noul B-arbore nu va mai avea spații nefolosite;

• se parcurge B-arborele în post-ordine și se creează un nou B-arbore, pornind direct de la ultimul nivel (al nodurilor frunză);

• de fiecare dată când un nod frunză se completează, următoarea cheie se introduce în nodul aflat pe nivelul imediat superior, și se continuă cu următorul nod frunză;

• același procedeu se aplică și pentru nodurile aflate pe nivele superioare.

Însăilarea unui B-arbore simplifică mult și operațiile de inserare și ștergere atunci când este nevoie să urcăm pentru a extinde sau a restrânge nodurile aflate pe nivelele superioare.

Schița unei implementări

Să presupunem că avem un fișier de date cu înregistrări de lungime variabilă, cheia fiecărei înregistrări este o valoare întreagă pe 32 de biți (între 1 și 4.294.967.295), și dorim să-l indexăm. În acest scop vom folosi B-arbori de ordin 39, și fiecare nod va avea structura următoare:

1 octet - gradul de ocupare a nodului(numărul de chei)
3 octet - pointer către nodul imediat superior (B-arborele este însăilat)
4 octeți - valoarea cheii înregistrării
4 octeți - poziția înregistrării în fișierul de date (față de începutul fișierului)
2 octeți - lungimea înregistrării în octeți
... ... - acest grup de zece octeți se repetă de 78 de ori
3 octeți - pointer către primul nod imediat inferior
... ... - în total 79 de pointeri

Un bloc cu această structură ocupă 1021 octeți, și dacă adăugăm încă 3 octeți se ajunge la 1024, care este un multiplu de capacitatea unui sector de disc, fapt care garantează un acces la noduri. Cei trei octeți suplimentari pot fi utilizați pentru a marca informații speciale: nodul curent este nod rădăcină, intermediar, frunză. Un nod va avea mărimea de1024 de octeți, indiferent de gradul de ocupare al acestuia.

Unele câmpuri nu au nici o semnificație:

- nodul rădăcină: câmpul pointer către nodul părinte;

- noduri frunză: câmpurile pointer către nodurile fii;

- un nod oarecare: dacă primul câmp are valoarea p, atunci au semnificație doar primele p grupuri de 10 octeți (valoare cheie, adresă și lungime înregistra-

Variațiuni

Există numeroase variante de B-arbori, care prezintă ușoare diferențe structurale față de modelul descris în acest articol, deși principiul rămâne același. Iată câteva posibilități:

Modelul descris în articol presupune unicitatea cheilor. Există însă și posibilitatea de indexare pe baza unor chei care nu sunt unice.

Așa cum se specifică și în articol, există posibilitatea de a "însăila" arborele, plasând în cadrul fiecărui nod o referință (pointer) la nodul părinte. Dacă arborele nu este însăilat, atunci toate operațiile trebuie implementate recursiv.

Există o variantă de B-arbore, numită **B+arbore** (*B+tree*) în care nodurile frunză sunt înlănțuite (de obicei, dublu) în ordinea cheilor. Această implementare este mult mai eficientă în cazul parcurgerii secvențiale (complete sau parțiale) a arborelui în ordinea (crescătoare sau descrescătoare) a cheilor, deoarece nu mai este necesară traversarea nodurilor de pe nivelurile superioare. Dezavantajul este că operațiile de inserare și ștergere devin mai complexe, mai cu seamă pentru nodurile frunză, deoarece se cere păstrată înlănțuirea.

O variantă numită **B*arbore** (*B*tree*) cere ca fiecare nod (exceptând rădăcina) să fie completat în proporție de cel puțin 2/3. Această condiție conduce la structuri mai compacte, rezultatul fiind o parcurgere mai rapidă. Dezavantajul este însă că operațiile de înserare și ștergere a cheilor devin mai complicate, deoarece redistribuirea cheilor în caz de depășire se face pornind de la două noduri (la inserare) sau de la trei noduri (la ștergere).

(M.S.)

re), și primii p+1 pointeri către nodurile imediat inferioare.

Pentru a simplifica accesul la B-arbore, se poate utiliza următoarea convenție: nodul rădăcină va fi întotdeauna pe prima poziție. Căutarea unei chei se poate face după următorul algoritm:

```
nodul curent este initial radacina;
repeta
  se cauta cheia data in nodul curent;
  rezulta: o adresa si
          un raspuns dupa cautare;
  daca se returneaza "Succes"
  atunci adresa returnata indica
          inregistrarea cautata;
       returneaza inregistrarea gasita;
  altfel adresa returnata indica
          nodul fiu;
       daca nodul fiu exista atunci
       nodul fiu devine nod curent;
la nesfirsit;
```

Funcția de căutare (bazată pe algoritmul de căutare binară) poate fi implementată astfel (presupunem că numărul de chei memorate în nodul curent este p):

```
l:=0; r:=p - 1;
repeta
  daca r < l atunci
      returneaza l si "Esec";
  m:=( l+r ) / 2;
  dif:=( Cheia_cautata - Cheia (m) );
  daca dif=0 atunci
      returneaza m si cc"Succes";
  daca dif < 0 atunci r:=m - 1
  altfel l:=m+1;
la nesfirsit
```

Bibliografie

Donald Knuth - *Tratat de programarea calculatoarelor. Algoritmi fundamentali*, Editura Tehnică 1976

Donald Knuth - *Tratat de programarea calculatoarelor. Sortare și căutare*, Editura Tehnică 1976.

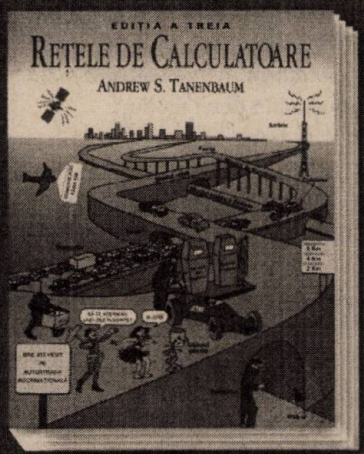
Ion Cozac este lector la Universitatea „Petru Maior" din Târgu-Mureș și poate fi contactat prin e-mail la adresa cozac@uttgm.ro.

O CARTE COMPUTER PRESS AGORA

Rețele de calculatoare

E timpul...

Prima carte românească de referință în domeniul rețelelor de calculatoare



Autor, profesor și cercetător, laureat al premiului ACM Karl V. Karlstrom Outstanding Educator Award, Andrew S. Tanenbaum explică cu grijă felul în care rețelele funcționează în interior, pornind de la tehnologia hardware până la cele mai utilizate aplicații de rețea. Cartea urmează o abordare structurată a rețelelor de calculatoare, începând de jos (de la nivelul fizic) și parcurgând treptat calea spre vârful acestora (nivelul aplicațiilor).

Subiectele abordate includ:

- Nivelul fizic (firele de cupru, fibrele optice, comunicațiile radio și cele prin satelit)
- Nivelul legătură de date (principiile protocoalelor, HDLC, SLIP și PPP)
- Subnivelul de acces la mediu (LAN-uri IEEE 802, punți, noile LAN-uri de mare viteză)
- Nivelul rețea (rutarea, controlul congestiei, interconectarea rețelelor, IPv6)
- Nivelul de transport (principiile protocoalelor de transport, TCP, performanța rețelei)
- Nivelul aplicație (criptografia, poștă electronică, știri, Web-ul, Java, multimedia)

În fiecare capitol sunt descrise în detaliu principiile de bază, urmate apoi de exemple ample luate din Internet, rețele ATM și rețele de comunicații fără fir.

Doresc să cumpăr cartea Rețele de calculatoare

V-am expediat în contul Computer Press AGORA nr.: 2021 400027100178 deschis la Banca Transilvania, filiala Mureș, 100.000 lei (în care nu sunt incluse taxele poștale), cu mandatul poștal/ordinul de plată nr.: ..... din data de ..... ☐ Doresc să-mi trimiteți și factură.

Pentru elevi și studenți cartea costă 75.000 lei. Ei vor trebui să atașeze comenzii o copie după carnetul de elev sau student.

Numele și prenumele: ..... Funcția: .....

Firma: ..... Adresa: .....

Cod poștal/Localitate: ..... Telefon/Fax: .....

Data ..... Semnătura .....

Despre DANGLING-ELSE

de Tiberiu Socaciu

Dacă v-ați "prins urechile" de câteva ori în ramurile unor condiții imbricate și n-ați mai înțeles cărui "IF" îi corespunde un anume "ELSE", atunci cu siguranță veți aprecia acest articol. Altfel, dacă veți citi totuși acest articol, veți putea evita multe ore pierdute cu depanarea programelor. Altfel... Unde rămăsesem?

Scopul prezentului articol este de a prezenta pe scurt Problema else-ului ambiguu (*dangling else*) precum și metodele de corectare a acestei "deficiențe".

O problemă

Ne propunem rezolvarea în limbajul Pascal (sau într-unul din limbajele asemănătoare) a următoarei probleme: efectuarea atribuirii `a:=a/b` dacă raportul din membrul al drept al atribuirii este definit. O "rezolvare" ar putea fi:

```
Program rezolvare;
var a,b:Real;
Begin
  Write('a=');
  Readln(a);
  Write('b=');
  Readln(b);
  if b<>0
  then
    if a<>0 {dacă a=0 nu fac nimic}
    then a:=a/b
  else writeln('Nu fac impartiri la 0!');
  Writeln('Noul a este:',a)
End.
```

La o testare a programului se obțin însă rezultate ce nu concordă cu așteptările.

Explicația

În Pascal, instrucțiunea `if` are una din următoarele două forme:

if *<condiție>*
then *<tratare-then>*
else *<tratare-else>*

respectiv:

if *<condiție>* **then** *<tratare-then>*

unde `<condiție>` este o expresie logică, iar `<tratare-then>` și `<tratare-else>` sunt instrucțiuni (în particular pot fi alte `if`-uri). Prima formă a instrucțiunii corespunde unei secvențe alternative complete, cu două ramuri, în timp ce a doua formă a instrucțiunii corespunde unei secvențe alternative cu o singură ramură. Prin urmare, secvențele de forma:

if c1 **then if** c2 **then** s1 **else** s2 (*)

pot fi interpretate semantic în două moduri:

if c1 **then** (**if** c2 **then** s1) **else** s2

sau

if c1 **then** (**if** c2 **then** s1 **else** s2)

În concluzie, ambiguitatea construcțiilor sintactice de tipul (*) este generată de existența a două structuri logice distincte care o generează. Acest fenomen se numește **dangling-else** (Pârv și Vancea: *Fundamentele limbajelor de programare*) și apare în multe limbaje (în special cele de tipul Pascal-like). Chiar și C-ul are acest defect, autorii limbajului recunoscând acest lucru în manualul limbajului (vezi Kernighan și Ritchie: *The C Programming Language*).

Rezolvarea

În principiu, rezolvările problemei sunt de diverse tipuri: soluții radicale (vezi Algol-60), moderate (Algol-68) și soluții de compromis (vezi PL/I, Pascal, C). Practic, dacă traducem în limbajul gramaticilor independente de context, atunci producțiile generatoare ale instrucțiunii `if`

(I) și ale altor instrucțiuni (S) ar fi:

S → I
I → if C then S
I → if C then S else S

a) **Cazul Algol-60 (Soluție radicală, modificarea instrucțiunii `if`)**

În Algol-60 s-a statuat că după `then` nu poate apare o altă instrucțiune `if` (T):

S → I
S → T
I → if C then T
I → if C then T else S

b) **Cazul Algol-68 (soluție moderată, modificarea instrucțiunii `if`)**

În Algol-68 instrucțiunile `if` trebuie să se termine cu un terminator de `if` (notat `fi`):

S → I
I → if C then S fi
I → if C then S else S fi

De remarcat că soluția a fost adoptată și de alte limbaje, cum ar fi Fortran-77, precum și de unele interpretoare de comenzi, cum ar fi shell-urile de sub Unix.

c) **Cazul PL/I (soluție de compromis, interpretare semantică)**

În PL/I rezolvarea ambiguității este de natură semantică și nu sintactică: un `else` "ține" de cel mai apropiat `if` care-l precede și nu are `else`. Astfel că

```
if c1 then if c2 then s1 else s2
```

se interpretează:

```
if c1 then (if c2 then s1) else s2
```

Această soluție a fost adoptată și de limbajele Pascal și C.

d) **Soluția cu `else` obligatoriu**

O soluție ar putea fi folosirea obligatorie a lui `else`,

Gramatici

alfabet: o mulțime (Σ) finită și nevidă de elemente, numite simboluri.
secvență: peste alfabetul Σ : o succesiune de simboluri consecutive ale alfabetului Σ
secvența concatenată: $z=xy$, unde $x=a1...an$ și $y=b1...bm$ reprezintă două secvențe peste alfabet, care concatenate dau naștere lui $z=a1...an\ b1...bm$
mulțimea tuturor secvențelor peste alfabetul Σ: se notează cu Σ^*
lungimea unei secvențe: numărul de simboluri care o alcătuiesc
secvența vidă (ε): secvența de lungime 0
un limbaj (L) peste alfabetul Σ: $L \subseteq \Sigma^*$
cuvinte : elementele unui limbaj
gramatică: cvadruplu $G=(N, \Sigma, P, S)$, unde
N - alfabetul simbolurilor neterminale
Σ – alfabetul simbolurilor terminale ($N \cap \Sigma = \phi$)
P - mulțimea producțiilor
(o producție se notează cu $\alpha \rightarrow \beta$ și se va citi α *se înlocuiește cu* β)
S – simbolul inițial.
derivare ($\Rightarrow$) : operație prin care un simbol neterminal este expandat în alte secvențe posibile, respectând regulile sintactice stabilite de producții
limbajul generat de o gramatică $G = (N, \Sigma, P, S)$: $L(G) = \{w \mid w \in \Sigma^*, S \Rightarrow w\}$. Altfel spus: mulțimea cuvintelor (formate din simboluri terminate) care pot fi obținute prin derivare (pe baza producțiilor), pornind de la simbolul inițial.
gramatică de tip 0: nu are nici o restricție referitoare la forma producțiilor sale
gramatică de tip 1 (dependentă de context): producțiile sale sunt de forma $\alpha A\beta \rightarrow \alpha\gamma\beta$ cu $\gamma \neq \varepsilon$, adică A se retranscrie ca γ numai în contextul generat de α și β
gramatică de tip 2 (independentă de context): producțiile sale sunt de forma $A \rightarrow \alpha$ cu $\alpha \neq \varepsilon$, (independent de contextul apariției sale, A se transcrie în α)

Exemplu de gramatică:
$G=(N, \Sigma, P, S)$, unde
$N=\{s, H\}$
$\Sigma=\{b, c, d, e\}$
$P=\{S \rightarrow bbSe \mid H,\ H \rightarrow cHdd \mid cd\}$

Limbajul generat de această gramatică:
$L(G)=\{b2\ n\ c2\ m\ d2\ m\text{-}1\ en \mid m \geq 1;\ n \geq 0\}$
Într-adevăr, pornind de la simbolul de start S, prima regulă de formare conduce la varianta $bbSe$, unde S, la rândul său poate fi dezvoltat fie sub aceeași formă, fie folosind producția $S \rightarrow H$. În acest moment există din nou două posibilități de urmat: $H \rightarrow cd$ sau se continuă derivarea: $H \rightarrow cHdd$.

adică neacceptarea construcțiilor de tipul al doilea (vezi paragraful "Explicația"):

S → I
I → if C then S else S

Probabil că o astfel de soluție este dezavuată de programatori din cauza "încorsetării" obligatorii de folosire a două variante alternative, chiar dacă una din ele are efect nul. Într-una din primele versiuni ale limbajului Fortran a fost definită instrucțiunea `if` logic:

if *<condiție> <etichetă-then> <etichetă-else>*

adică ceva foarte aproape de această soluție...

Iată cum se rezolvă inconsistența (*) în acest fel:

```
if c1
then
  if c2
  then s1
  else { instructiune vida }
else s2
sau

if c1
then
  if c2
  then s1
  else s2
else { instructiune vida }
```

e) Paranteze **begin..end**

Din cauza restricției din Algol-60, în cazul în care după `then` apăreau totuși secvențe `if`, acestea trebuiau închise între parantezele `begin` și `end`. Redăm rezolvarea inconsistenței (*):

```
if c1
then
  begin
    if c2 then s1
  end
else s2
sau
if c1
then
  begin
    if c2 then s1
          else s2
  end
```

Chiar dacă nu este foarte plăcută la folosire (părere proprie), este la ora actuală cea mai folosită soluționare (în limbaje de tipul Pascal-like) a dangling-else-ului. Probabil că foarte mulți dintre noi programăm așa fără să ne punem prea multe întrebări, sau poate din automatism. Totuși, nu strică să conștientizăm din când în când pericolul...

Încă o problemă

Se cere să se afișeze numele fetelor care au vârsta mai mare de cinci ani, precum și numele băieților care au vârsta peste șase ani.

Prezentăm mai jos implementarea algoritmilor în Pascal și în Fox. Prima variantă este cu **else** vid, cea de-a doua include **if**-ul în cauză între **begin** și **end**.

```
Program exemplu; { folosind else vid }
type copil=record
             nume:string[10];
             varsta:Byte;
             sex:Char
           end;
var a:array[1..10] of copil;
    n,i:Integer;
Begin
   Write('Numarul copiilor:'); Readln(n);
   for i:=1 to n do
     with a[i] do
     begin
       Write('nume='); Readln(nume);
       Write('varsta='); Readln(varsta);
       Write('fata/baiat (F/M) ='); Readln(sex)
     end;
   for i:=1 to n do
     with a[i] do
       if Upcase(sex)='F'
       then
         if varsta>=5
         then Writeln(nume:10,varsta:10,sex:2)
         else   { instructiune vida }
       else
         if varsta>=6
         then Writeln(nume:10,varsta:10,sex:2)
End.

Program exemplu;{ folosind paranteze begin...end }
type copil=record
             nume:string[10];
             varsta:Byte;
             sex:Char
           end;
var a:array[1..10] of copil;
    n,i:Integer;
Begin
   Write('Numarul copiilor:'); Readln(n);
   for i:=1 to n do
     with a[i] do
     begin
       Write('nume='); Readln(nume);
       Write('varsta='); Readln(varsta);
       Write('fata/baiat (F/M) ='); Readln(sex)
     end;
   for i:=1 to n do
     with a[i] do
```

```
if Upcase(sex)='F'
then
  begin
    if varsta>=5
    then Writeln(nume:10,varsta:10,sex:2)
  end
else
  if varsta>=6
  then Writeln(nume:10,varsta:10,sex:2)
End.
```

Datorită faptului că în Fox se specifică sfârșitul unei instrucțiuni **if** prin **endif**, rezolvarea problemei va avea o singură variantă.

```
close all
set talk off
clear
n = 0
dimension nume(10), varsta(10), sex(10)
@ 1,1 say 'Introduceti numarul copiilor:'
  @ 1,35 get n
read
for i = 1 to n
  nume(i) = space(10)
  varsta(i) = 17
  sex(i) = 'F'
endfor
for i = 1 to n
  @ 3*i,1 say 'nume='
    @ 3*i,25 get nume(i)
  @ 3*i+1,1 say 'varsta='
    @ 3*i+1,25 get varsta(i)
  @ 3*i+2,1 say 'fata/baiat (F/M) ='
    @ 3*i+2,25 get sex(i)
  read
endfor
for i = 1 to n
  if sex(i) == 'F'
    if varsta(i) >= 5
      ? nume(i), varsta(i), sex(i)
    endif
  else
    if varsta(i) >= 6
      ? nume(i), varsta(i), sex(i)
    endif
  endif
endfor
wait
```

În loc de final

Prezentul articol este elaborat pe baza unui material nepublicat din 1992. Nu am pretenția de a fi epuizat subiectul, scopul fiind doar de a atrage atenția asupra acestui fenomen. Soluțiile enumerate sunt clasice. În cazul în care descoperiți una mai interesantă, vom fi bucuroși să citim un nou articol în GInfo.

Tiberiu Socaciu este licențiat și master al Facultății de Matematică și Informatică a Universității "Babeș-Bolyai" Cluj-Napoca. Lucrează ca inginer de sistem la S.C. SAR Transilvania S.A. Poate fi contactat prin e-mail: infodata@theoffice.net.

PROGRAME autoverificabile

de Cosmin TRUȚA

Oricărui programator i s-a întâmplat să nu-i funcționeze programul conform așteptărilor sale. Se pot pierde ore întregi pentru depistarea unei erori ascunse cine știe unde. Uneori aceasta trece neobservată peste toate testele dar își arată colții în momentele cele mai nepotrivite: când profesorul vine să verifice programul, când clientul este aproape convins că aplicația este utilă pentru el sau...

Experiența ne-a învățat că este o utopie ideea de a realiza programe complexe, care nu necesită nici cea mai mică corectură. Calculatorul - ca orice componentă a mediului înconjurător - se supune legilor lui Murphy. Conform uneia dintre aceste legi (*Calculatoarele fac ceea ce le spunem să facă, nu ceea ce vrem să facă*) fiecare încearcă în felul său să minimizeze cel puțin pagubele produse de erori. Practici precum *scrierea foarte atentă a codului, demonstrarea teoretică a corectitudinii algoritmilor folosiți* sau *testarea folosind baterii voluminoase de teste* sunt frecvent întâlnite. În ciuda tuturor măsurilor de precauție, întrebări de genul *Dar cine este generalul Protection Fault?!*, *Ce caută Division by Zero aici?*, *Din nou Abnormal Program Termination??* apar și nu ne înspăimântă foarte tare, deoarece suntem obișnuiți cu ele așa cum ne-am obișnuit cu fel de fel de îndemnuri, de la "Please contact your vendor" până la "Press Ctrl+Alt +Del to continue".

Nu vom prezenta o soluție completă a problemei expuse dar, dacă veți ține cont de conținutul articolului, veți scăpa de multe bătăi de cap de acum încolo. Înainte de a începe aș vrea să fac o mică observație tehnică: se vor bucura de avantajele metodei cei care folosesc limbajul C, C++ sau un alt limbaj în care sistemul de macrodefiniții este suficient de flexibil. Alte limbaje - precum EIFFEL - care au înglobat un mecanism de specificare a precondițiilor, a postcondițiilor și a invarianților, ușurează și mai mult scrierea programelor *bug-free*. În cazul celorlalte limbaje (PASCAL, JAVA etc.) se pot face mici artificii ce vor suplini lipsa suportului oferit de limbaj.

Problemă rezolvată - sursa altor probleme?

Să presupunem că avem de scris o funcție care calculează rădăcina pătrată a unui număr real nenegativ. Vom folosi algoritmul iterativ:

$a_1 = 1$, $a_{n+1} = (a_n + (x / a_n)) / 2$, cu $a_n \to sqrt(x)$ dacă $n \to \infty$ și $x > 0$.

Rapid, scriem următorul cod:

```
double my_sqrt(double x)
{  double x1=1,x2;
   if (x==0)
     return 0;
   while (1)
   {  x2=(x1+x/x1)/2;
      if (fabs(x1-x2)<=EPS)
        return x1;
      x1=x2;
   }
}
```

Deoarece formula este valabilă doar pentru valori strict pozitive, situația când x este nul am tratat-o separat. Cu toate acestea, dacă se apelează funcția cu un parametru negativ, se va intra într-o buclă infinită. Ar fi normal să scriem o condiție de ieșire din funcție în cazul în care x e negativ cu returnarea unui cod de eroare, însă decidem să nu facem acest lucru (!!). Funcția este scrisă doar pentru uz intern. Presupunem că nu va fi apelată niciodată cu un parametru negativ (eventual facem testul înainte de apelul funcției). Vom considera că este inutil să supraîncărcăm codul cu o condiție în plus, adică o variabilă de eroare returnată ca parametru, o variabilă globală a cărei valoare să fie testată ulterior sau lansarea unei excepții, dacă se folosește limbajul C++.

Situații în genul celei prezentate sunt frecvente. Amintiți-vă doar de câte ori ați efectuat operații cu pointeri fără a verifica dacă ei sunt valizi (nenuli), din simplul motiv că verificarea ar aduce ramificări incomode și inutile programului: cu puțină atenție - nu se intră în funcție cu date incorecte.

Totuși, pe neașteptate programul crapă! "Aaaa... uite care era cauza!!" Adică am ajuns cu execuția pe unde nu trebuie, din cauză că *am uitat* să facem niște inițializări sau niște validări și astfel unele variabile au primit valori

nepermise. și mai dificil este atunci când vizual, efectul apare la mult timp după ce datele nu mai sunt consistente. Pe de altă parte, a insera la fiecare pas condiții redundante de testare a consistenței datelor cu tratări speciale nu se practică din motive de eficiență, de lizibilitate și nu în ultimul rând, de evitare a altor erori, provenite din apariția unor ramificații foarte greu de urmărit și testat.

Aserțiuni, precondiții, postcondiții

O soluție elegantă pentru rezolvarea situației conflictuale expuse este *aserțiunea*. Ea constă într-o linie de cod care enunță presupunerea făcută asupra stării programului la un moment dat. Mai concret, în exemplul prezentat vom scrie prima instrucțiune, imediat după declarația variabilelor:

```
assert(x>=0);
```

cu mențiunea că, la începutul fișierului sursă vom plasa directiva

```
#include <assert.h>
```

În cazul unui program C++, aserțiunea poate fi chiar prima instrucțiune din corpul funcției, deoarece limbajul nu impune declararea variabilelor chiar la începutul funcției. Dacă folosim un compilator ce respectă specificațiile noului standard ANSI/ISO C++, în loc să includem antetul `assert.h` vom folosi `#include <cassert>` ce va avea același efect.

Interpretarea aserțiunii de mai sus este următoarea: *presupunem că x>=0*. Ne putem gândi la această interpretare ca la ceva mai mult decât un comentariu: instrucțiunea nu va avea nici un efect asupra variantei finale a programului, dar în timpul dezvoltării și depanării, ea va avea ca efect oprirea execuției programului cu un mesaj de genul:

```
Assertion failed: n>=0,file
my_sqrt.c,line 8
Abnormal program termination
```

Vă veți întreba probabil de ce am ales o astfel de soluție din moment ce înainte am vorbit despre faptul că nu avem de gând să oprim execuția programului dacă o condiție oarecare nu este satisfăcută, cu afișarea unui mesaj inestetic și fără nici un înțeles pentru utilizator.

Ei bine, nu este același lucru! Gândiți-vă că în timpul dezvoltării aveți setate opțiuni de compilare care inhibă optimizările și produc informații adiționale pe lângă fișierul binar executabil, utile pentru depanare. în schimb, când programul a ajuns în varianta finală, adică *este gata*, se elimină opțiunile ce produc informațiile pentru depanare și se activează optimizările, pentru a produce cod fie mai rapid, fie mai scurt dar, bineînțeles cu același efect asupra datelor de intrare. Aserțiunile sunt niște *gardieni* care lasă execuția programului în pace dacă sunt respectate condițiile impuse ca argument, dar o întrerup în caz contrar, cu afișarea unui mesaj referitor la condiția încălcată și la locul în fișierul sursă unde lucrurile nu sunt tocmai în regulă. Gardienii respectivi *dispar* atunci când aplicația este pregătită pentru livrare, ea putând rula la viteza maximă.

Veți înțelege mai bine modul lor de funcționare după ce vă voi da un exemplu de implementare a lor, conform standardului ANSI C. Puteți consulta fișierul antet `assert.h` al compilatorului dumneavoastră!

```
#ifdef NDEBUG
#define assert(cond) ((void)0)
#else
#define assert(cond)
((cond)? (void)0 :
(void)__assertfail
(#cond, __FILE__, __LINE__))
#endif
```

unde `__assertfail(char *cond, char *file, int line)` este o funcție de bibliotecă ce provoacă terminarea anormală a programului, cu afișarea la dispozitivul standard de eroare a mesajului ce avertizează încălcarea condiției `cond`. Implementarea macroului `assert` se mai poate întâlni și sub forma:

```
#ifdef NDEBUG
#define assert(cond) ((void)0)
#else
#define assert(cond) \
  do
  { \
  if (!cond)
  { \
  fprintf(stderr,
  "Assertion failed:
  %s file %s line %d\n",
  #cond, __FILE__, __LINE__); \
  abort(); \
  } \
  } while (0)
#endif
```

Stilul puțin întortocheat în care sunt scrise macrodefinițiile permit utilizarea pseudo-instrucțiunii `assert` ca și cum ea ar fi orice altă instrucțiune simplă. Sunt permise chiar și construcții de genul:

```
if (x!=0)
  assert(y==x);
```

În standardul ANSI C, simbolul `NDEBUG` are o semnificație specială: dacă este definit înainte de începerea compilării unui text sursă, mai exact înaintea includerii antetului `assert.h` (sau `cassert` în ANSI C++), atunci instrucțiunile `assert` nu vor avea nici un efect. Altfel ele vor testa condiția specificată și vor permite continuarea normală a execuției programului doar dacă acea condiție este îndeplinită.

În concluzie, a presăra prin cod aserțiuni la momente potrivite este o bună practică. În acest fel vor fi semnalate din timp orice fel de inexactități a căror posibilă existență a fost prevăzută. Ele pot contribui chiar la creșterea clarității codului, deoarece dacă le vom întâlni în textul sursă al unei funcții, vom observa mai repede ce condiții trebuie să fie îndeplinite la apelul funcțiilor sau ce condiții vor respecta datele de ieșire. Când considerăm că nu mai este necesar să avem codul binar executabil încărcat cu condiții redundante, nu avem decât să definim simbolul `NDEBUG` fie prin opțiunile compilatorului, fie prin plasarea unei directive `#define` într-un fișier antet propriu care să fie inclus de către toate modulele aplicației ca apoi să recompilăm programul.

Iată cum va arăta exemplul nostru rescris:

```
double my_sqrt(double x)
{  double x1=1,x2;
   assert(x>=0);
   if (x==0)
     return 0;
   while (1)
   {  x2=(x1+x/x1)/2;
      if (fabs(x1-x2)<=EPS)
      {  assert(
           fabs(x-x1*x1)<=EPS)
         return x1;
      }
      x1=x2;
   }
}
```

Principiile auto-verificării au fost introduse de către Bertrand Meyer în *Object-Oriented Software Construction*. Aici el a introdus noțiunea de *precondiție* (prima aserțiune din exemplu), de *postcondiție* (cea a doua aserțiune, înainte de ieșirea din funcție) și de *invariant* și a descris limbajul de programare EIFFEL, limbaj în care toate acestea sunt implementate nativ. O precondiție a unei proceduri, funcții, metode etc. trebuie să fie adevărată la intrare. Ea verifică de obicei corectitudinea parametrilor și a datelor globale ce au influență asupra execuției procedurii. O postcondiție descrie rezultatul funcției. Ea se aplică la ieșire și este exprimată în funcție de datele de intrare. În practică se obișnuiește să se scrie doar postcondiții simple, atunci când starea datelor de intrare a fost modificată și nu mai este disponibilă la ieșire.

În C și C++ precondițiile și postcondițiile sunt implementate prin intermediul aserțiunilor. Mai există încă un mecanism de auto-verificare - *invariantul* - însă nu ne vom ocupa de el acum.

Să nu exagerăm

Este foarte important să nu pierdem din vedere următorul lucru: *aserțiunile sunt utile atunci când vrem să ne asigurăm de inexistența unui eveniment care nu trebuie să apară, singura cauză a eventualei sale apariții fiind o eroare de programare*. Aceasta înseamnă că situațiile de excepție care pot apare în sistemul final și care nu pot fi evitate (imposibilitatea de a deschide un fișier, terminarea memoriei disponibile, introducerea greșită a datelor de către utilizator, și multe altele) le veți trata în modul în care v-ați obișnuit. în nici un caz să nu scrieți de pildă, un cod de căutare a unui element într-o listă liniară ca și cel de mai jos:

```
while (lista->val!=elem->val)
{
  assert(lista->urm != NULL);
  /** greșit! **/
  lista = lista->urm;
}
```

Implementări specifice

Se poate să nu fiți de acord cu constrângerea de a defini simbolul NDEBUG pentru eliminarea codului aserțiunilor. Dacă vi se pare mai comod să definiți o opțiune de compilare suplimentară nu pentru varianta finală a aplicației (release) ci pentru varianta de lucru (debug), scrieți următoarele linii într-un fișier antet care va fi inclus în fiecare modul, înainte de a include assert.h / cassert:

```
#ifdef NDEBUG
#undef DEBUG
#else
#ifndef DEBUG
#define NDEBUG
#endif
#endif
```

Astfel, aserțiunile vor fi compilate dacă este definit simbolul DEBUG. Dacă DEBUG nu este definit se va defini automat simbolul NDEBUG. Pentru a nu permite existența simultană a ambelor simboluri s-a făcut un test suplimentar.

Se cuvine să vorbim puțin și despre modalitatea în care firma Microsoft a implementat propriul său mecanism de aserțiuni. Este vorba despre implementarea din biblioteca **MFC** (Microsoft Foundation Classes) din mediul Visual C++ unde avem în principal două simboluri: ASSERT(cond) și VERIFY(cond). Deosebirea dintre ASSERT și assert constă în faptul că dacă aserțiunea este falsă nu se întrerupe programul ci se afișează o fereastră de dialog în mediul Windows cu mesajul corespunzător urmată de apelul depanatorului din mediul integrat. Astfel se permite inspectarea variabilelor pentru a determina cauza erorii și se permite de asemenea continuarea execuției programului.

Pe lângă funcția macro ASSERT mai există una - VERIFY - care funcționează întocmai ca ASSERT, cu deosebirea că expresia evaluată e întotdeauna compilată, chiar și în variantele non-depanare deși testarea expresiei nu mai are loc în versiunile finale. Cu alte cuvinte macroul VERIFY este util în situații ca:

```
VERIFY(FunctieNenula()!=0);
```

unde FunctieNenula trebuie apelată, iar rezultatul său trebuie să fie nenul. Dacă nu se folosește VERIFY în loc de ASSERT, sau assert în varianta release a programului, apelul funcției este complet eliminat. Efectul lui VERIFY este similar cu cel prezentat în liniile următoare:

```
int tmp=FunctieNenula();
ASSERT(tmp!=0);
/* sau assert(tmp!=0); */
```

Ca ultime mențiuni referitoare la implementarea Microsoft pot spune că există alte simboluri (ASSERT_VALID, ASSERT_POINTER, ASSERT_KINDOF etc.) al căror efect îl puteți afla consultând documentația bibliotecii MFC.

Iată un artificiu simplu prin care putem folosi aserțiuni și în alte limbaje. Exemplul următor scris în limbajul PASCAL oferă două alternative: dacă limbajul pe care îl folosiți nu suportă deloc preprocesarea (cum ar fi JAVA) puteți alege alternativa care nu folosește directivele preprocesorului.

```
{$IFDEF DEBUG}
const debug=true;
{$ELSE}
const debug=false;
{$ENDIF}
procedure assert(cond:Boolean;
const msg:String)
begin
  if not cond
  then
  begin
```

```
    Writeln('Assertion
      Failed: ', msg);
    Halt(3);
  end;
end;
...
{ o modalitate: }
{ se folosesc directive }
{ de preprocesare }
{$IFDEF DEBUG}
  assert(x=1, 'x=1');
{$ENDIF}
...
{ alta modalitate: }
{ nu se folosesc directive }
{ de preprocesare }
  if (debug)
    assert(x=1, 'x=1');
...
```

Deoarece debug are o valoare constantă compilatorul va putea face optimizări, în sensul că va elimina efectiv din codul executabilului apelul procedurii assert, atunci când debug are valoarea false. Această remarcă este valabilă pentru orice limbaj care dispune de un compilator performant.

Bibliografie

Accredited Standards Committee, *November 1996 Working Paper*, ANSI/ISO C++ Draft Standard, 1996

•••

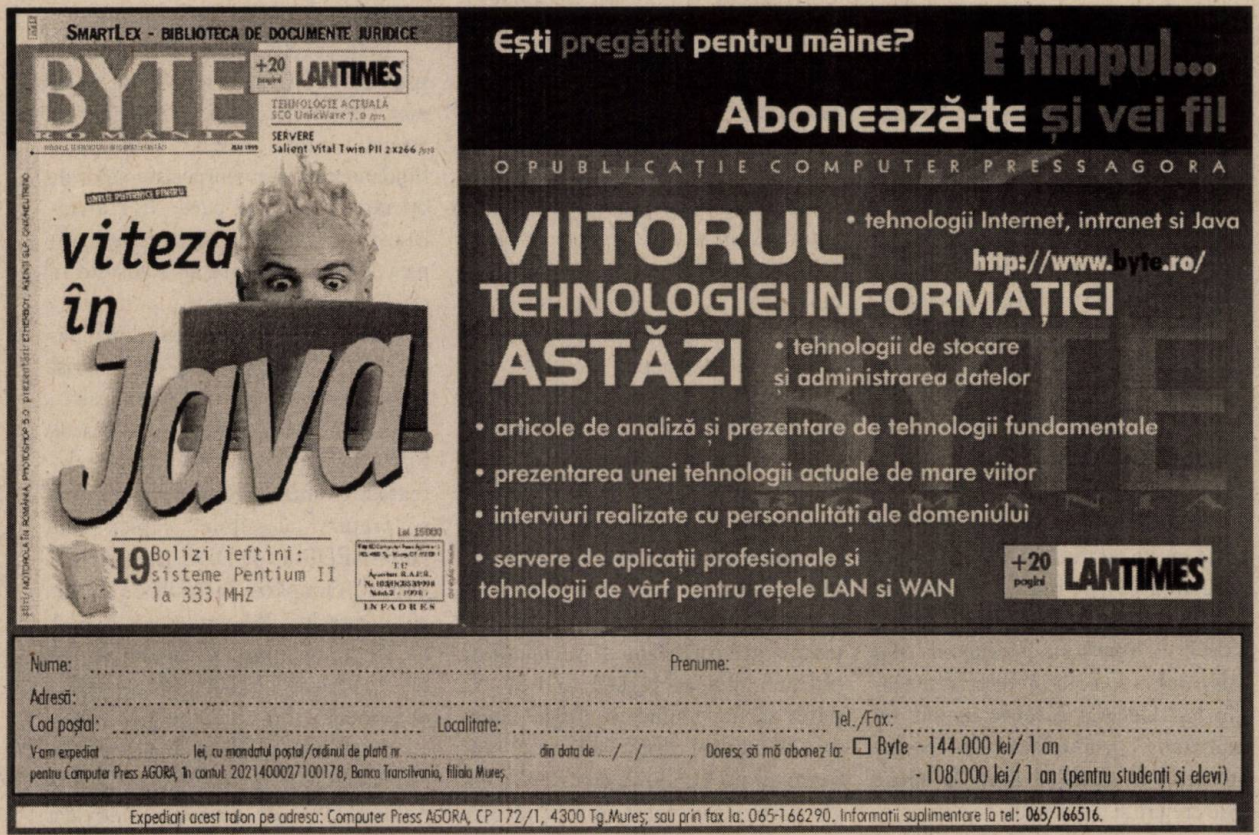
Frenţiu Militon, *Verificarea corectitudinii programelor*, Curs, Universitatea "Babeş-Bolyai", Cluj-Napoca, 1997

•••

Weir Charles, *Code That Tests Itself, C++ Report*, Vol.10 / No.4 (April 1998), pag.26-31

Williams Mickey, *Bazele Visual C++*, Editura Teora, 1997

Dl. Cosmin Truţa este absolvent al Facultăţii de Matematică şi Informatică, Secţia Informatică, din cadrul Universitatăţii "Babeş-Bolyai", Cluj-Napoca. E-mail: cosmint@cs.ubbcluj.ro



UNIX - scurtă istorie

de Mihai Budiu

Orice informatician profesionist care s-a întâlnit cu Unix odată s-a îndrăgostit cu siguranță de el. Motivele pasiunilor pe care le suscită constituie ele însele un subiect despre care s-au scris sute de cărți și nu despre asta intenționez să povestesc aici. Nu despre "de ce" voi vorbi, ci mai curând despre "cum" și "când". Istoria genezei acestui sistem de operare este ea însăși plină de învățăminte interesante.

Putem coborî pe firul timpului pentru a regăsi rădăcinile Unix până la zilele de început ale calculatoarelor, prin anii '40—'50. Atunci lumea (de fapt "selecta" comunitate a informaticienilor) încerca să construiască un set de "scule" care să mărească productivitatea calculatoarelor, care pe atunci costau o grămadă de bani și-și petreceau tare mult timp citind cartele sau făcând alte treburi de același gen - plictisitoare și mari consumatoare de timp - scule care vor fi sămânța a ceea ce mai târziu se va numi "Sistem de Operare". Ar fi însă prea departe o astfel de întoarcere, deși cu siguranță nu lipsită de interes.

Povestea începe cu adevărat prin 1962, când trei mari companii din domeniul calculatoarelor din Statele Unite și-au unit forțele pentru a da naștere unui monstru software. MIT (*Massachusets Institute of Technology* - o universitate de numele căreia sunt legate multe pietre de hotar din istoria calculatoarelor), Bell Laboratories (care pe vremea aceea aparținea de AT&T - *American Telephone and Telegraph* - un colos al telecomunicațiilor) și General Electric (atunci un important producător de calculatoare) s-au decis să pună la bătaie niște creiere și o grămadă de bani ca să facă un "utilitar" sofisticat. Acesta trebuia să permită folosirea unei capacități de calcul în comun de către sute de utilizatori, cu un maximum de eficiență și securitate. Modelul la care aspirau era cel al rețelei de electricitate: vroiau ca fiecare potențial utilizator să aibă la dispoziție un fel de mufă la care să cupleze un terminal și prin care să aibă acces instantaneu la toate resursele de calcul ale sistemului. Vroiau să aibă un singur calculator care să satisfacă toate nevoile utilizatorilor din Boston. Proiectul s-a numit MULTICS, prescurtare de la *MULTiplexed Information and Computer Service* - serviciu multiplexat la informații și putere de calcul. Acesta este unul dintre cele mai răsunătoare eșecuri din istoria informaticii mondiale.

Proiectul n-a ajuns la sfârșit. Bell s-a retras după câțiva ani și câteva milioane de dolari cheltuiți. General Electric a renunțat cu totul să mai lucreze în industria calculatoarelor.

Motivele eșecului sunt multe. Unul dintre ele este puterea hardware-ului de atunci. Ideea că peste douăzeci de ani un calculator de puterea celui pe care îl foloseau va putea să fie vândut cu numai câteva mii de dolari drept un calculator personal nu era bună nici ca SF. Un alt motiv, paradoxal, a fost o prea mare efervescență de idei. Fiecare din cei implicați era un artizan, și varietatea conceptelor a dus la o proastă integrare a părților. Nu în ultimul rând a contribuit limbajul folosit: MULTICS ar fi trebuit să fie scris în PL/I, iar compilatorul era dezvoltat în același timp de o altă echipă, care niciodată nu era în termen.

Retrospectiv vorbind e greu de spus că MULTICS a fost un mare eșec. Majoritatea ideilor elaborate atunci sunt acum conceptele fundamentale din teoria sistemelor de operare. Pe de altă parte MIT a realizat până la urmă o variantă funcțională MULTICS care se folosește în câteva locuri din lume.

MULTICS a fost prima dintre rădăcinile Unix. O a doua a fost fenomenala dezvoltare a minicalculatoarelor. De la niște coloși (foarte, foarte scumpi), o "rasă" de calculatoare a evoluat spre dimensiuni mici și prețuri accesibile. Tipică este familia PDP a lui DEC (numit apoi Digital, cumpărat în 1998 de Compaq), care în 1961 vindea un PDP-1 cu 4K de memorie RAM de câte 18 biți cu numai 120.000$ (de douăzeci de ori mai ieftin decât un IBM 7094). La prețul ăsta PDP-urile se vindeau ca pâinea caldă. La asta s-a adăugat și politica DEC de a scoate o întreagă familie de calculatoare (PDP) compatibile unele cu altele.

Faptele

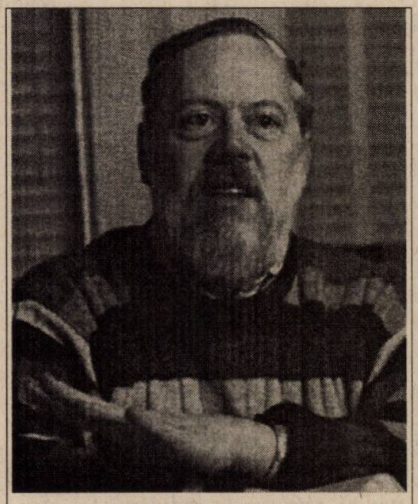
Unul dintre participanții la proiectul MULTICS din partea Bell Labs, Ken Thomson, după nefericita experiență, a găsit prin sediul firmei un PDP-7 pe care aparent nimeni nu-l folosea. Și-a propus să scrie pe el un program foarte interesant, și anume un joc de simulare a unor nave spațiale și a sistemului solar. Calculatorul avea tot hardul necesar, dar soft de loc. Așa că Thomson s-a apucat de lucru folosind întâi un *cross-assembler* (un asamblor care se executa pe o mașină dar genera cod pentru alta). După ce s-a plictisit să tot care benzi cu hârtie găurita de la asamblor la PDP ca să încarce programele s-a hotărât să-și scrie un asamblor pentru PDP. Zis și făcut.

Să scrii un joc însă nu e o treabă deloc banală. E o provocare pentru orice programator, oricât de experimentat. Jocurile sunt programe extrem de complexe, care implică adesea algoritmi sofisticați. Pe deasupra sunt greu de depanat și testat. Ei, asta-i trebuia de fapt și lui Thomson: un *mediu de dezvoltare*, care să-i permită să scrie ușor și eficient programul. Atunci el și-a folosit cunoștințele acumulate cu prilejul MULTICS și a scris, împreună cu un coleg, Dennis Ritchie, o variantă mai sărăcuță a acestuia pentru PDP. Prima piesă a fost un sistem de fișiere, după care au urmat un subsistem pentru execuția proceselor și apoi un interpretor de comenzi (*shell*). Colegii lui au numit rezultatul, ca să râdă de Thomson, UNICS (*UNiplexed Information...*). Uite însă că numele a prins, deși grafia a fost apoi schimbată în Unix. Această primă variantă era scrisă în limbajul de asamblare al mașinii PDP-7.

Programul lui Thomson i-a fermecat pe colegii lui, programatori și ei. Iată însă că unul din departamentele hârțogare ale Bell avea nevoie de niște programe mai de doamne-ajută pentru prelucrarea de texte. Sarcina a căzut tot pe umerii lui Thomson, care a primit pentru aceasta un calculator mai mare și mai frumos, numit PDP-11/20. Si pentru că decât să scrie direct programele mai simplu i-ar fi fost să aibă un mediu de programare asemănător pe noua mașină, s-a decis să "porteze" Unix pe noua arhitectură. Compatibile, compatibile, dar nu chiar de tot. Totul trebuia rescris de la cap, in limbaj de asamblare PDP-11.

Un nou limbaj

Aici interesele lui s-au unit cu al altui grup de la Bell, care lucra la niște limbaje de programare care pot exprima algoritmi mai precis decât engleza și mai fiabil decât FORTRAN-ul. Ei elaboraseră deja două noi limbaje care vor avea un rol important în evoluția ulterioară a domeniului. Acestea erau Algol și BCPL. Thomson și-a scris un interpretor de B - o versiune proprie de BCPL, dar acesta era mult prea lent ca să poată fi folosit pentru programe mari. Denis Ritchie a fost însărcinat să scrie un compilator pentru B. Ritchie era și el un tip foarte inventiv, așa ca a modificat masiv limbajul pentru a-l face cit mai eficace fără a pierde însă *portabilitatea*. Programele nu trebuiau să facă prea multe presupuneri despre arhitectura mașinii pe care rulează, pentru a se potrivi pe cât mai multe dintre ele. Dialectului creat de el, Ritchie i-a spus C.



Dennis Rithie

C a fost limbajul potrivit la momentul potrivit. Este totodată simplu (sub 40 de cuvinte rezervate!), flexibil, extrem de eficient (aproape de mașină!), structurat, rapid și portabil. De altfel, de atunci încoace a dominat categoric scena programării de sistem: majoritatea sistemelor de operare, compilatoarelor și utilitarelor existente în lume sunt scrise în C. Succesul C a depășit cu mult pe cel al Unix-ului. C este încă direct sau indirect limbajul dominant în lume și la ora actuală (indirect prin urmașii săi C++ și Java).

Thomson și Ritchie au rescris Unix în C pentru noile mașini, și apoi pe succesoarele lor PDP-11/45 și PDP-11/70. Acestea din urmă aveau mecanisme de protecție hardware și au permis o implementare eficientă a mecanismelor de securitate din Unix.

Primele versiuni de Unix au fost folosite numai în interiorul Bell Laboratories. Ediția a treia, din februarie 1973 includea un compilator de C. În același an întregul Unix a fost rescris în C. Thomson și Ritchie au fost co-autori ai unui articol acum faimos, numit "The Unix Time-Sharing System", publicat în *Communications of the ACM* în iulie 1974.

O nouă gazdă

Prima portare în afara familiei PDP a fost făcută pe un calculator Interdata 8/32 [Notă: Investigații istorice au arătat că prima portare a fost de fapt făcută la Universitatea Wollongong din Australia, urmată la scurtă durată de o portare făcută de Bell Labs.]. Aceasta a fost ocazia de a constata că Unix se baza câteodată pe anumite particularități ale mașinii pe care trebuia să ruleze (cum ar fi că adresele sunt pe 16 biți, deci sunt maximum 64K de memorie). O grămadă de timp a fost petrecută pentru a face Unix independent de asemenea idiosincrazii.

La început transpunerea pentru Interdata s-a desfășurat lent, pentru că dezvoltarea se făcea pe PDP, care se afla la etajul 5 al clădirii. Interdata era la etajul 1. O nouă versiune însemna să compilezi la etajul 5, și să cobori cu benzile la 1, să vezi dacă merge. După ce s-au uzat scările, a apărut un amplu interes pentru posibilitatea conectării a două mașini

printr-un mediu electronic. Acesta a fost începutul programelor de comunicație în rețea de calculatoare din Unix.

Sistemul era foarte ușor acum de portat, pentru ca 95 la sută din el era scris in C. Dacă vroiai să-l muți pe o mașină nouă, tot ce aveai de făcut era să scrii un compilator de C pentru mașina aceea, iar un compilator e mult mai mic decât un întreg sistem de operare. Asta nu era chiar totul, pentru că pe noua mașină trebuia să mai scrii un pic cod în asamblare (întreruperi, porturi) și să rescrii partea care se ocupa cu perifericele (driverele pentru discuri, benzi, terminale). Oricum, treaba era mult mai simplă pentru că tot ceea ce rămânea putea fi transportat neschimbat, iar asta e mare lucru! (Câte din programele scrise de voi se compilează ușor pe un calculator complet diferit de cel pe care le-ați dezvoltat?)

Lucrurile nu s-au oprit aici! Steve Johnson de la Bell s-a apucat să studieze compilatorul de C și a constatat că o mare parte din compilare se face la fel independent de mașina pentru care se generează codul. Ideea lui a fost să parametrizeze compilatorul în așa fel încât să-i furnizezi doar o descriere a mașinii pentru care trebuie să genereze cod, iar grosul compilatorului să fie neschimbat. Acesta a fost compilatorul PORTABIL de C. Ani întregi compilatoarele de C de peste tot au fost bazate pe cel al lui Johnson, și aceasta a ajutat mult la răspândirea Unix-ului.

Versiunea cea mai importantă de Unix numită "Versiunea 7" a fost lansată în ianuarie 1979; era primul sistem Unix cu adevărat portabil, mult mai robust și mai funcțional decât toate versiunile anterioare.

Versiuni de UNIX

nume	bazat pe	fabricant
AIX	SV	IBM
A/UX	SV	Apple
BOS	SV	Bull
BSD	BSD	University of California at Berkeley (UCB)
Chorus	SV	Chorus
Coherent	SV	Mark Williams Company
EP/IX	POSIX	Control Data Corporation
FreeBSD	BSD	UCB
GNU Hurd	Mach	Free Software Foundation
HP-UX	SV+BSD	Hewlet Packard
IRIX	SV	Silicon Graphics
Linux	POSIX	Linus Torvalds + hackeri
Minix	clone SV	Tanenbaum (Vrije Universitet Amsterdam)
Net/2	BSD	UCB
NEWS-OS	?	Sony
QNX	?	Quantum Software
OSF/1	Mach	DEC, Open Software Foundation
Plan 9	-	AT&T
SCO Xenix	SV	SCO
SCO Unix	SV	SCO
Solaris	SV	Sun Microsistems
SunOs	BSD	Sun Microsistems
Ultrix	BSD	DEC
UNICOS	SV	Cray
UnixWare	SV	Novell
UTEK	?	Tektronix
Xenix	SV	Microsoft

Propagarea

Poate nici unul din motivele de mai sus nu ar fi fost suficient pentru ca Unix să cunoască popularitatea de care se bucură azi. A mai fost ceva. Pe vremea aceea AT&T era un monopol pe telecomunicații in SUA. Or americanii au niște legi destul de severe anti-trust (vedeți și procesul intentat de departamentul de justiție împotriva companiei Microsoft.) AT&T semnase cu DOJ (*Department of Justice*) o înțelegere prin care se abținea să activeze în industria calculatoarelor. De aceea n-a avut nimic împotrivă să vândă licența codului sursă Unix, pentru prețuri modice, universităților. PDP-11 era, printr-o fericită coincidență, foarte răspândit în universități, dar sistemul de operare cu care era livrat de DEC nu prea era pe placul utilizatorilor. Unix a fost providențial. Cursurile erau ținute pe Unix, cu exemple luate din Unix. Codul se putea studia la seminare. Lumea era într-o efervescență nesfârșită (un informatician obsedat este foarte obsedat!). Fiecare inova ceva, fiecare depana ceva. Conferințe, întruniri, comunicări. Un profesor australian, John Lions, a scris pe marginea codului un comentariu de genul celor pentru operele lui Shakespeare sau Chaucer. Versiunile se succedau versiunilor.

În 1975 sistemul deja rula în colțuri foarte îndepărtate ale lumii: Universitatea Berkeley din California, universitatea Ebraică din Ierusalim, universitatea South Wales din Australia și universitatea din Toronto din Canada. Studenții absolvenți au dus apoi cu ei în industrie lecțiile învățate cu Unix, contribuind la răspândirea lui în continuare. AT&T încorpora adesea contribuții din întreaga lume în codul sistemului (colaborare care s-a deteriorat simțitor când AT&T a început mai târziu să comercializeze sistemul).

În 1984, Thomson și Ritchie au primit prestigiosul premiu Turing al ACM (*Association for Computing Machinery*) pentru lucrarea lor din 1974 care descria pentru prima oară Unix. ACM este o asociație profesională a calculatoriștilor din întreaga lume, de un prestigiu rivalizând cu IEEE.

Lucrurile păreau a fi într-o deplină concordie, dar nu a fost să rămână așa.

Divergența

În 1979 Microsoft (sic!) împreuna cu Santa Cruz Operations (SCO) au portat sistemul Unix pentru PCuri cu 80x86, sub numele de XENIX, una dintre primele variante comerciale ale sistemului.

Cam în acea perioadă, Thomson a avut un an sabatic (în sistemul american un cercetător cu experiență din industrie face o dată la câțiva ani o pauză și lucrează - de obicei - în învățământ și invers) la Universitatea Berkeley din California. Berkeley era una dintre primele universități care avusese licența codului. Studenții și cercetătorii de la Berkeley au dezvoltat un sistem propriu gen Unix numit BSD (*Berkeley Software Distribution*) și care avea o mulțime de îmbunătățiri față de System V.

Printre părțile tari BSD: un mecanism de memorie virtuală cu paginare, inspirat din sistemul de operare VMS de la Digital, scris de Bill Joy și Ozap Babaoglu, care permitea executarea unor programe mai mari decât încăpeau în memorie, un sistem de fișiere mult mai rapid ("The Fast File System"), compilatoare de LISP și Pascal și o implementare a protocolului de comunicații pentru rețele de calculatoare numit TCP/IP, care este de departe cel mai răspândit astăzi, mult în afara lumii Unix (este protocolul folosit în rețeaua Internet), plus o grămadă de utilitare de toate felurile. Armata a investit o grămadă de bani în cercetarea legată de Unix la Berkeley, prin agenția DARPA (*Defense Advanced Research Projects Agency*) [Notă: O mare parte din cercetarea tehnologică în Statele Unite este sponsorizată de DARPA.]. Dezvoltarea lui BSD a fost continuată la Berkeley până în 1993. În 1982, Bill Joy a plecat de la Berkeley pentru a înființa propria lui companie numită Sun Microsystems, care a lansat o variantă proprie de Unix numită SunOS, bazată pe BSD.

Tot în 1982 legea americana a spart AT&T în mai multe companii independente și concurente pentru a sfărâma monopolul asupra telecomunicațiilor. AT&T a avut voie să intre în arena calculatoarelor. Prima mișcare: a schimbat clauzele licenței și a făcut din el un sistem comercial, pentru care trebuia să dai o grămadă de bani și ale cărui surse nu erau public disponibile. Prima variantă s-a numit System III, urmată curând după aceea de System V. SV (cum mai este numit) avea multe facilități și inovații față de BSD (memorie partajată, semafoare, o nouă arhitectură de memorie virtuală, biblioteci partajate, *streams*, etc.).

Spre sfârșitul anilor '80 în lume existau două stiluri mari de Unix: BSD și System V, care adesea nu prea se potriveau între ele. Acest lucru împiedica succesul comercial al produselor, pentru că nu puteai garanta că programele vândute vor funcționa pe orice variantă.

Marile firme de calculatoare au început să elaboreze propriile variante, cu tot felul de adaosuri și modificări. Unele dintre ele se bazează pe codul cumpărat de la AT&T și dezvoltat, altele sunt scrise de la cap la coadă, altele împrumută idei de la BSD. Fiecare mare fabricant de calculatoare are propriul sau Unix:

Firmă	Unix
Hewlet_Packard	HP UX
IBM	AIX
Sun	SunOS, Solaris
Digital	Ultrix
Silicon Graphics	IRIX
SCO	SCO Unix

Numai pentru calculatoarele din familia PC există mai multe Unix-uri mititele: XENIX, Coherent, 386BSD, Minix, Linux, etc.

Standardizarea

Datorită proliferării de variante nu mai era foarte clar ce e și ce nu e Unix. Fiecare fabricant adăuga propriile lui "ornamente". Sistemele bazate pe BSD și SV erau diferite într-o mulțime de aspecte fundamentale.

Din păcate încercarea de a standardiza Unix a dus la apariția unui întreg set de standarde incompatibile! În final 3 standarde diferite s-au bucurat de o acceptare largă: SVID, POSIX și X/Open. Standardele descriu numai ce fel de servicii trebuie să exporte sistemul de operare, și nu modul în care acestea sunt implementate.

Primul standard a fost făcut public de AT&T și a fost numit *System V Interface Definition*, sau SVID (1983). Acesta este practic o descriere precisă a tuturor funcționalităților pe care SV le oferea. AT&T a publicat și o serie de teste pe care un sistem trebuia să le treacă înainte de a putea fi numit "Unix".

Oficiul de standarde al IEEE (Asociația Inginerilor Electroniști) a

Linus Torrvalds, creatorul Linux-ului

făcut un alt pas spre reconciliere. Sute de experți din industrie, universități și guvern au format un colectiv numit POSIX (*Portable Operating System - unIX*).

Standardul elaborat de ei este publicat într-o serie de documente numite seria POSIX 1003. Ele spun explicit cum trebuie să arate un sistem de operare ca să fie un Unix.

Spre deosebire de standardele obișnuite din calculatoare, acesta este deosebit de bine scris, și a extras din System V și BSD intersecția calităților iar nu reuniunea lor, plus câteva din lucrurile foarte bune din BSD. Prima versiune a fost publicată în 1990.

În plus, limbajul C a fost standardizat de ANSI (*American National Standard Institute*).

În fine, X/Open este o coaliție de

fabricanți și comercianți de calculatoare care a extins specificațiile POSIX cu o grămadă de alte recomandări, de pildă despre sistemele de ferestre sau managementul datelor. Recomandările lor sunt publicate sub numele X/Open Portability Guide.

Bariera sonică

La mijlocul anilor '80, la Universitatea Carnegie Mellon a început un proiect de cercetare în arhitectura sistemelor de operare, folosind un nou sistem de operare numit Mach

[Notă: Ernst Mach este un mare fizician din secolul 20; 1 Mach este și unitatea de măsură a vitezei egală cu viteza sunetului; de aici și titlul acestei secțiuni.].

Mach era un sistem revoluționar, micro-nucleu, care oferea o serie foarte simplă de servicii peste care se puteau construi ca din cărămizi de Lego oricare altele; de exemplu Mach putea executa Unix ca pe un proces și putea oferi toate serviciile Unix, simultan cu ale unui alt sistem de operare. Mach nu avea nici un fel de restricții de licențiere ca Unix, așa că a devenit extrem de popular; multe sisteme comerciale se bazează pe Mach 2.5: OSF/1, despre care vom vorbi mai jos, și NextSTEP, al companiei NEXT, fondată de Steve Jobs (cunoscut ca fondator al firmei Apple). Sistemul de operare Windows NT (1992) de la Microsoft are o arhitectură extrem de apropiată de Mach.

Războiul

În 1987 AT&T a cumpărat 20% din acțiunile Sun și a intrat într-un parteneriat strâns cu Sun pentru a dezvolta noile versiuni de Unix. AT&T a anunțat că Sun se va bucura de tratament preferențial în ceea ce privește accesul la surse; Sun a renunțat complet la nucleul SunOS, bazat pe BSD și a îmbrățișat cealaltă versiune, SV. AT&T și Sun au lansat împreună System V Release 4, sau SVR4. Sistemul de operare de la Sun este numit de atunci încoace Solaris 2.x.

Ca reacție, un grup de mari fabricanți, cuprinzând IBM, Digital, Hewlet Packard, Apollo a fondat un consorțiu numit OSF (*Open Software Foundation*) care lupta împotriva dominației AT&T și Sun în arena Unix. OSF era finanțată de membrii componenți și trebuia să dezvolte o gamă largă de software și să extindă sistemul Unix în alte direcții; unele din părțile dezvoltate sunt: un sistem de ferestre grafice (X11, dezvoltat la MIT), o interfață grafică (Motif), un mediu de calcul distribuit (DCE - *Distributed Computing Environment*), un mediu de management distribuit al resurselor (DME).

Primul sistem de operare al OSF a fost lansat de Digital s-a numit OSF/1 și era bazat pe o combinație de Mach, 4.3, BSD și anumite funcțiuni din nucleul de la IBM, AIX. În plus oferea o sumedenie de avantaje tehnologice care nu se găseau în oferta de la Sun/AT&T (suport pentru procesoare multiple, biblioteci dinamice, etc.)

Ca să se răzbune, AT&T, Sun și alți comercianți care se bazau pe SVR4 au făcut un consorțiu numit Unix International (UI).

Decăderea

Când doi se ceartă al treilea câștigă. O scădere economică la începutul anilor

CRONOLOGIE

1968 Proiectul Multics
1969 Unics pentru PDP-7, Bell Labs, AT&T
1971 Unix versiunea 1, asamblor PDP11/20
1974 Unix versiuna 4, C
1975 Unix versiunea 6, foarte răspândit
1978 2.x Berkeley Software Distribution (BSD)
1978 3BSD, memorie virtuală
1979 Unix V7 "ultimul Unix adevărat"
1980 4.0BSD
1982 System III, Unix comercial de la AT&T
1982 Fondarea Sun Microsystems
1983 System V
1983 4.2BSD, TCP/IP
1984 System V Release (SVR)2; publicată SV Interface Definition SVID
1984 X/Open, consorțiu de comercianți Unix
1985 SVID1, SVR2
1986 SVR3, SVID2
1986 4.3BSD
1986 Mach
1987 X/Open Portability Guide 2 publicat
1987 Alianță AT&T - Sun
1988 SVR4, amestecă SV, BSD, SunOS
1988 Se fondează OSF
1989 X/Open Portability Guide 3
1991 OSF/1 bazat pe SVR2; micronucleu Mach 2.5
1992 Linux e creat
1992 X/Open Portability Guide 4
1992 Windows NT 1.0
1993 4.4BSD; dezvoltarea Unix la Berkeley încetează
1993 Novell cumpără Unix System Laboratories
1993 Novell vinde dreptul pentru marca Unix lui X/Open
1995 Digital OSF/1
1996 GNU Hurd alfa lansat
1998 Solaris 2.6
1998 Windows NT 5.0 (amânat)

'90, rapida evoluție a performanței PC-urilor și dominanța crescândă a sistemului de operare Windows al Microsoft au oprit creșterea și au amenințat supraviețuirea Unix. UI a fost desființată în 1993 iar OSF și-a îngustat simțitor anvergura planurilor. (Chiar și sistemul OSF/1 a fost redenumit în 1995 Digital Unix).

A urmat o serie rapidă de schimbări de proprietar pentru marca comercială Unix și drepturi asupra codului. În 1991 Novell a cumpărat o parte din drepturile asupra Unix de la AT&T și a lansat propria lui versiune numită UnixWare în 1992. În 1993 Novell a cumpărat ce mai rămăsese de la AT&T, iar Sun a cumpărat dreptul de a folosi cod de la Novell. De curând SCO a cumpărat în întregime Unix de la Novell.

Supraviețuirea?

Lupta marilor companii pentru dominanță a dus aproape la dispariția Unix-ului. Dar un contracurent foarte spectaculos și-a făcut la apariția la început pe nesimțite, dar apoi cu din ce în ce mai multă forță.

Lucrurile au pornit de la un informatician american care lucrează la universitatea "Frei" din Amsterdam, Andrew Tanenbaum. În 1987 el a publicat codul unui mic sistem de operare, care pe dinafară seamănă cu Unix, deși pe dinăuntru este complet diferit (într-un stil micro-nucleu, ca Mach). Sistemul lui Tanenbaum se numea MINIX și putea fi folosit fără nici un fel de plată în scopuri academice, dar cu restricții de copiere, modificare și distribuire. Codul merge pe o varietate de calculatoare și este excelent comentat. (O a doua ediție a cărții și MINIX 2.0 au fost publicate în 1997.)

Comunitatea utilizatorilor MINIX se bucura de o creștere constantă când, în 1992, un student finlandez în vârstă de 22 de ani, Linus Torvalds, după lectura cărții în care Tanenbaum dă sursele sistemului lui, modifică inițial MINIX, scrie un sistem pentru PC cu 80386, și îl face "free software", adică fără restricții de copiere și distribuire. Faptul că era ceva mai performant, datorită design-ului tradițional, a dus la o răspândire explozivă a sistemului, care are la ora actuală milioane de utilizatori în întreaga lume.

La versiunea actuală de Linux (2.1.10x) au pus umărul mii de programatori pasionați din întreaga lume. Fiecare a găsit un *bug*, a scris un utilitar, a scris o documentație. În aceste condiții codul Linux este de foarte bună calitate (eficient și fără bug-uri grave) rivalizând și surclasând implementări de firmă. În plus este absolut gratuit.

Colaborarea aceasta a fost posibilă datorită dezvoltării extraordinare a rețelelor de calculatoare. La crearea produsului final au contribuit mii de oameni care nu s-au întâlnit niciodată! Lipsa restricțiilor și disponibilitatea codului sursă face Linux un candidat ideal pentru cercetare și dezvoltare. La ora aceasta, Linux este considerat de analiști singurul sistem de operare care poate oferi o alternativă solidă dominației totale Microsoft. Cele mai mari firme de calculatoare renunță încetul cu încetul la variantele lor de Unix speriate de creșterea rapidă a Windows NT. Deși Windows NT este încă un sistem relativ imatur față de Unix, forța comercială a Microsoft precum și compatibilitatea cu alte sisteme Windows, care domină 95% din piață sunt niște adversari de temut.

Pentru comparație, iată dimensiunile nucleelor în linii de cod-sursă:

Nucleu	Linii
Linux	1 milion
Solaris	11 milioane
Windows NT	40 milioane

Pe de altă parte, Linux se bucură de o creștere constantă, sub directa supervizare a autorului său, Linus, care acum lucrează la o companie nou înființată în California. Linux este pe departe cel mai portat sistem de operare din istorie, funcționând pe o gamă extrem de largă de calculatoare și stații de lucru, și pentru care din ce în ce mai multe companii lansează software.

Încheiere

Sistemele Unix comerciale sunt încă niște realizări impresionante, care oferă funcționalități superioare lui Linux sau NT.

Unii analiști cred că versiunile de Unix vor dispărea încet din arena jucătorilor "grei" economici; ultimul supraviețuitor comercial pronosticat fiind Sun. Mulți pariază pe o luptă Linux/NT. Ce se va întâmpla însă numai un profet ar putea spune. Oricum, deși substanțial sub cifra de afaceri a lui Microsoft, sistemele Unix continuă să reprezinte o piață de mai multe zeci de miliarde de dolari.

La ora actuală Unix este singurul sistem de operare care funcționează simultan pe microcalculatoare și supercalculatoare. Este practic singurul sistem de operare pentru minicalculatoare și stații de lucru. TCP/IP este protocolul folosit pe cea mai mare rețea de calculatoare din lume: Internet, cu sute milioane de abonați pe toate continentele.

Eleganța interfețelor sale și calitatea remarcabilă a abstracțiilor pe care le oferă utilizatorilor au influențat și vor continua să influențeze profund arhitectura sistemelor de calcul.

Să ne amintim că totul a început de la un joc.

NOTE BIBLIOGRAFICE

Informațiile din acest text sunt extrase dintr-o multitudine de surse. Iată aici unele dintre ele:

- Andrew S. Tanenbaum, "*Modern Operating Systems*", Prentice Hall, 1995.
- Uresh Vahalia, "*Unix Internals*", Prentice Hall, 1995.
- Thomson, K., Ritchie, D. "*The UNIX Time-Sharing System*", Communications of the ACM, vol 17, no 7, iulie 1974.

Mihai Budiu este doctorand la Carnegie Mellon University, SUA.

E-mail: mihaib@cs.cmu.edu.

Web: http://www.cs. cmu.edu/mihaib.

EXCEPȚII în Java

de Horia Georgescu

Este cu siguranță foarte greu (poate chiar imposibil) să prevedem toate situațiile care pot interveni la rularea unui program. Cu toate acestea, sarcina proiectanților de software este să creeze programe care să facă față în aproape toate cazurile. Unele limbaje, printre care și Java, oferă mecanisme specializate pentru tratarea situațiilor nedorite.

Noțiunea de excepție

Un aspect important de care trebuie ținut cont ori de câte ori scriem un program este de a asigura robustețea sa. Prin aceasta înțelegem să prevedem toate situațiile nedorite (de exemplu introducerea unor date eronate, eșecul la încercarea de a avea acces la Internet etc.) și a specifica pentru ele acțiunile corespunzătoare.

Există însă și reversul aspectului descris mai sus. Introducerea de verificări prin instrucțiuni de control va face programul greu de citit și va afecta buna sa structurare.

Ceea ce dorim este să avem un mecanism care:

- să permită ca fiecare condiție legată de robustețea programului să fie verificată exact acolo unde poată să apară;
- atunci când condiția apare (suntem în situația că s-a întâmplat ceva nedorit), controlul să fie trecut automat unei secțiuni de cod aflată în program "într-un loc potrivit" (care poate fi departe de locul unde condiția a fost detectată), cod care realizează acțiunea corespunzătoare ce trebuie întreprinsă. Acest cod va fi numit *handler* (mânuitor) *de excepție*.

Limbajul Java rezolvă problema de mai sus printr-un mecanism numit *excepție*. Excepția este o indicație că a intervenit o anumită condiție specială (nedorită); spunem că segmentul de cod care a detectat condiția *lansează o excepție* de tipul corespunzător. Lansarea unei excepții atrage după sine *captarea* ei de către un handler de eroare; în lipsa acestuia programul se va încheia cu un mesaj de eroare.

Mai precis, trebuie să realizăm următoarele:

- să definim tipurile de excepție care să permită lansarea excepțiilor dorite;
- să lansăm o excepție la "locul potrivit" în program;
- să prevedem la "locul potrivit" un handler de excepție, care să capteze excepția și să întreprindă acțiunile corespunzătoare.

O excepție este în fapt o instanțiere a unei clase derivate din `java.lang.Exception`; rezultă că *o excepție este un obiect*, care are câmpurile sale, constructorii săi și metodele sale.

Când un program detectează o condiție care cere ca o excepție să fie raportată, el creează un obiect corespunzător excepției (folosind constructorul adecvat) și îl lansează.

Un prim exemplu

Exemplul se referă la metoda `System.in.read()` care lansează, atunci când are loc o eroare la citire (nu poate fi citit un caracter), o excepție de tipul `IOException`. Există două metode de a trata această situație:

1) Se precizează, în antetul metodei care execută apelul respectiv, clauza:

```
throws IOException
```

după lista de parametri a metodei. Drept urmare ia naștere un obiect de tipul `IOException` care tipărește un mesaj de eroare și întrerupe executarea programului; putem gândi că este executată prin lipsă o metodă care întreprinde aceste acțiuni.

2) Se prevede o acțiune explicită specificată de programator prin:

```
try {
   ... System.in.read();...
}
catch (IOException e){
   acțiune
}
```

prin care este creat obiectul `e` de tipul

IOException, care poate fi folosit în prelucrările prevăzute de *acțiune*. Acțiunea respectivă poate consta, de exemplu, în tipărirea unui mesaj edificator, atribuirea unor valori prin lipsă, invocarea unor metode (presupunând că acestea există) ale clasei IOException prin intermediul obiectului e etc.

În această a doua variantă nu mai este necesar să declarăm clauza **throws** IOException în antetul metodei. De asemenea trebuie observat că nu mai este întreprinsă acțiunea prin lipsă specificată mai sus, deoarece este prevăzut explicit un cod ce trebuie executat și anume cel din **catch**.

De exemplu, acțiunea poate fi:

```
System.out.println("Eroare : " + e)
```

prin care este tipărit mesajul "Eroare" urmat de numele excepției; reamintim că semnul de concatenare +, în situația când unul dintre operanzi este de tip **String**, face ca ceilalți operanzi să fie convertiți la șiruri de caractere.

Clasa Exception are un constructor "prin lipsă" cu zero argumente, un constructor cu un argument de tip **String** și o metodă fără nume care se execută prin lipsă la lansarea unui obiect și care constă în tipărirea unui mesaj de eroare specific (la care se poate adăuga șirul transmis constructorului, în cazul în care fost folosit acest constructor).

Clauza throws și instrucțiunea throw

O excepție este lansată folosind instrucțiunea **throw**, a cărei formă este:

```
throw new tip_exceptie (parametri);
```

Efectul ei constă în crearea unei excepții (unui obiect) ca instanțiere a clasei tip_exceptie ce extinde clasa Exception. Crearea este însoțită de invocarea acelui constructor al clasei tip_exceptie, care este identificat prin numărul și tipul parametrilor din invocare.

De obicei lansarea unei excepții se face din cadrul unei instrucțiuni **if** sau a unei instrucțiuni **switch**, ca de exemplu în:

```
tip Met(...)throws Exceptie {
    ...
    if(...) throw Exceptie (parametri);
    ...
}
```

unde Exceptie este o clasă ce extinde clasa Exception.

Dacă este îndeplinită condiția din **if**, atunci este creat, folosind un constructor, un obiect având tipul (de excepție) Exceptie; acest constructor poate executa acțiuni asupra parametrilor transmiși, acțiuni menite să prelucreze excepția conform dorințelor programatorului.

Observăm că în metoda Met de mai sus apare clauza **throws** Exceptie.

Să remarcăm că mecanismul seamănă cu crearea obișnuită a unui obiect:

ELEMENTE

Continuăm în această coloană prezentarea elementelor de bază ale limbajului Java, prezentare începută în GInfo nr. 2/1998, pag. 31. Pentru "reîmprospătarea" memoriei, în acel număr au fost prezentate elemente de limbaj legate de ***caractere, identificatori, constante, comentarii, variabile*** *și* ***operatori****. (puteți găsi această prezentare în pagina de web www.ginfo.ro/8_2)*

Instrucțiuni

Limbajul Java prevede mai multe tipuri de instrucțiuni:

Instrucțiuni de declarare (declarări): declară o variabilă ca având un anumit tip și eventual îi atribuie o valoare inițială; variabila devine *locală* blocului respectiv, în sensul că există atâta timp cât se execută instrucțiuni ale blocului. Ele pot apare oriunde în interiorul unui bloc. Înainte de utilizarea lor, variabilele locale trebuie să fi primit valori fie prin inițializare la declarare, fie printr-o instrucțiune de atribuire (în caz contrar va fi semnalată o eroare la compilare).

Invocări de metode, indiferent dacă ele întorc sau nu o valoare.

Incrementări, care folosesc forma prefixată sau postfixată a lui ++ și --.

Instrucțiuni de atribuire, cele care conțin semnul = eventual prefixat cu un operator.

Instrucțiuni prin care sunt create obiecte: cele care folosesc **new** pentru a crea un nou obiect.

Instrucțiunea compusă, care grupează mai multe instrucțiuni prin includerea lor între acolade; ia naștere astfel un bloc. Un bloc este considerat a fi o unică instrucțiune.

Instrucțiunile ce controlează ordinea de executare, ca de exemplu: **if-else**, **for**, **while** etc. Ele vor fi prezentate detaliat în continuare.

Toate aceste instrucțiuni, cu excepția instrucțiunii compuse, se termină cu punct și virgulă.

Trecem la descrierea instrucțiunilor ce controlează ordinea de executare:

Instrucțiunea if-else

Are una dintre formele:

```
if (expresie_booleană) instr1;
```

sau

```
if (expresie_booleană) instr1;
else instr2;
```

Este evaluată expresia booleană. Dacă

```
import Exceptie;
class ...
  tip Met(...) {
   ...
    if(...) Exceptie
            Ob = new Exceptie(parametri);
    ...
  }
}
```

Lucrurile nu sunt însă identice: nu există constructorii de care am vorbit mai sus și nu au loc automat tipărirea unui mesaj de excepție și terminarea programului.

Sunt importante următoarele precizări:

• precizarea tipului de excepție în clauza **throws** este importantă pentru că programatorul (ca și compilatorul) trebuie să știe ce excepție poate fi lansată și care este tipul ei, la fel cum trebuie să cunoască tipul valorii întoarse de o metodă;

• dacă din interiorul metodei dorim să prevedem lansarea mai multor excepții, atunci toate tipurile lor trebuie precizate în clauza **throws**, despărțite prin virgulă;

• o metodă poate lansa numai excepțiile care sunt precizate în interiorul său (mai precis în clauza **throws**); lansarea poate avea loc din interiorul metodei, sau din interiorul altei metode pe care ea o invocă;

• nu este permis să lansăm excepția din interiorul inițializărilor statice; în schimb acest lucru este permis din interiorul constructorilor.

Construcția try

Mecanismul incipient descris mai sus permite crearea unei excepții (obiect) și invocarea unui constructor al său. Tipul de excepție respectiv este însă o clasă, care poate avea și metode. Dar obiectul creat nu are nume, deci deocamdată nu putem invoca aceste metode; nu este nici clar de unde am putea să le invocăm.

Construcția **try** are forma:

```
try bloc
catch(E_1 e_1) bloc_1
catch(E_2 e_2) bloc_2
...
finally bloc_final
```

unde:

`E_1,E_2,...` sunt tipuri de excepție;

`e_1, e_2,...` sunt identificatori;

clauza **finally** este opțională;

blocurile presupun includerea lor între acolade.

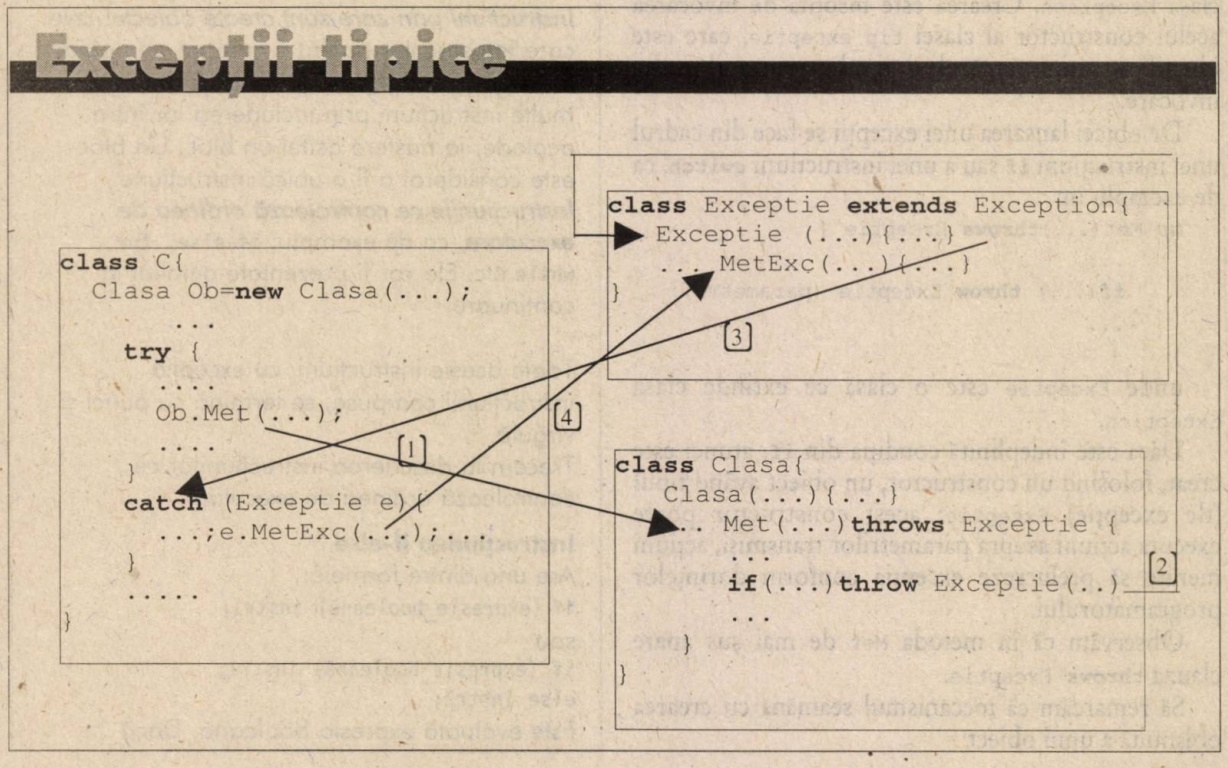
Vom presupune deocamdată că nu apare clauza **finally**.

Ideea de bază este următoarea:

dacă în interiorul blocului *bloc* nu s-a lansat nici o excepție, atunci se trece la sf(rșitul construcției;

dacă în interiorul blocului s-a lansat o excepție, atunci înseamnă că a luat naștere (după cum știm) un obiect. În continuare este identificat tipul de excepție `E_i` corespunzător și controlul este transferat clauzei **catch**`(E_i e_i)` *bloc_i*. În acest moment numele obiectului (excepției) poate fi referit cu `e_i` în interiorul blocului `bloc_i` ceea ce permite ca din

Excepții tipice



interiorul acestui bloc să fie lansate metode ale lui E_i. Spunem că excepția a fost captată de clauza **catch** corespunzătoare.

Clauza **catch** poate fi deci gândită ca o metodă inclusă, care are un parametru ce identifică excepția care a fost captată.

Descriem în continuare modul tipic de utilizare a excepțiilor. În figura "Excepții tipice" am considerat, pentru simplificare, un singur tip de excepție, cu un unic constructor și o unică metodă.

La construcția clasei Clasa, programatorul a identificat o condiție deosebită care ar putea interveni și de aceea a conceput și clasa Exceptie pentru a putea prevedea acțiunile ce trebuie întreprinse. În clasa Clasa, metoda Met este cea care poate lansa o excepție corespunzătoare acelei condiții. În mod normal, aceste activități sunt întreprinse o singură dată, în vederea unor aplicații ce urmează a fi dezvoltate. De aceea este natural ca cele două clase să se afle în fișiere separate.

Ulterior, programatorul scrie un program care cuprinde clasa C. Din interiorul construcției **try** este apelată metoda Met a clasei Clasa. În cazul în care apare condiția în discuție, este creat un obiect de tipul Exceptie prin invocarea constructorului său. Apoi controlul este trecut clauzei **catch** în interiorul căreia obiectul creat poate fi referit prin e, ceea ce permite invocarea metodei MetExc din Exceptie.

Acum discuția de la începutul capitolului capătă consistență: o excepție de tipul Exceptie este lansată din clasa Clasa și este captată în clasa C.

Vom aplica cele de mai sus la rezolvarea următoarei probleme:

Problemă: *Un vector* a *de lungime* n *conține* n *caractere. Ele trebuie mutate într-o ordine oarecare în vectorul* b, *prin intermediul unui vector auxiliar* stack *de lungime* lung, *care implementează o stivă. Operațiile permise sunt*:

- *mutarea unui element din* a *în vectorul* stack : *este mutat elementul curent (cu numărul de ordine cel mai mic) din* a *după elementele deja existente în stivă*;
- *extragerea (inclusiv ștergerea) unui element din vârful stivei și înscrierea sa în* b, *după ultimul element înscris în* b.

Dificultatea problemei constă în tratarea cazului în care se încearcă extragerea unui element din stiva vidă și a cazului în care se încearcă introducerea unui element în stiva plină.

Trecem la descrierea programului.

Clasa principală este Stiva. Metoda sa main folosește, în afară de vectorii a și b, următoarele:

- câmpul na semnificând poziția din a de pe care va fi mutat un element în stivă;
- câmpul nb semnificând poziția din b pe care va fi înscris următorul element;

rezultatul este true, atunci se execută instr1, iar în caz contrar se execută instr2 (sau nimic pentru forma scurtă).

Clauza **else** este asociată celui mai recent **if** căruia nu i s-a asociat un **else**. Astfel în :

```
if (a==0) b=1; if (c==1) b=2; else b=3;
```

clauza **else** este asociată celui de al doilea **if**. Pentru a face lucrurile clare, este indicat ca fiecare **if** să fie însoțit de un **else** (urmat eventual de instrucțiunea vidă).

Instrucțiunea while

Are forma:

```
while (expresie_booleană) instr;
```

Se începe cu evaluarea expresiei booleene și se execută în mod repetat instrucțiunea instr atâta timp cât expresia booleană are valoarea true.

Instrucțiunea do-while

Are forma:

```
do instr while(expresie booleană);
```

Se începe cu executarea instrucțiunii instr și se reia executarea ei atâta timp cât expresia booleană are valoarea true.

Instrucțiunea for

Are forma:

```
for
(initializari;expresie_booleana;incrementari)
instrucțiune;
```

unde:

- instrucțiunile și incrementările sunt materializate prin secvențe de instrucțiuni separate prin virgulă; secvența poate fi și vidă;
- expresia booleană poate și ea lipsi, caz în care se presupune că ea are (prin lipsă) valoarea true.

Efectul instrucțiunii poate fi descris astfel:

```
initializari;
while (expresie_booleana) {
      instructiune; incrementari;
}
```

Cadrul tipic de utilizare este în iterații, ca de exemplu:

```
s=0;
for(i=0;i<=n;i++) s+=i;
```

Mai prezentăm un exemplu: pentru inversarea ordinii elementelor unui vector a este suficientă o instrucțiune **for**:

```
for(i=0,j=a.]ength-1;i<j;i++,j--)
      { b=a[i];a[i]=a[j];a[j]=b;}
```

Un caz limită este:

```
for( ; ;) instr;
```

care desemnează repetarea "la infinit" a executării lui instr. Este de presupus că folosirea acestei forme este însoțită de o

• obiectul R de tipul Random (clasă din java.util), care va fi folosit pentru apelarea metodei nextInt cu 0 argumente, ce întoarce în mod aleator un număr întreg;

• obiectul Ob de tipul s, unde clasa s implementează operațiile asupra stivei. Clasa s conține metodele pune și ia. Metoda pune are un argument de tip **char** și este gândită pentru adăugarea acestui argument în stivă. Metoda ia nu are nici un argument și este gândită să întoarcă elementul din vârful stivei.

Metoda main din Stiva repetă, atâta timp cât în b nu au fost înscrise n caractere, următoarele acțiuni:

• este generat aleator un nou număr întreg;

• dacă valoarea sa este negativă și mai există elemente în a, se invocă metoda pune din clasa s cu intenția de a muta un element din a în stivă; invocarea este urmată de mărirea lui na cu o unitate. În caz contrar este invocată metoda ia cu intenția de a obține un element din stivă și de a-l muta în vectorul b. În final sunt listate elementele lui b.

Observăm că nu sunt tratate explicit cazurile de excepție (adăugare la stivă plină sau extragere din stiva vidă), aceste verificări căzând în sarcina clasei s. De asemenea utilizarea clasei Random permite un nedeterminism în succesiunea de operații asupra stivei, ceea ce va conduce la vectori b diferiți de la o executare la alta.

Trecem la descrierea clasei s. Este folosit vectorul stack de lungime lung pentru implementarea stivei. În ns este memorat numărul de elemente din stivă. Metoda pune are declarat în antet tipul de excepție plina, căruia îi corespunde clasa plina ce extinde clasa Exception. Deosebim două situații:

• dacă stiva este plină (ns=lung), atunci prin **throw** este creată o excepție (un obiect de tip plina). Controlul este trecut acelei clauze **catch** din Stiva care este identificată prin tipul plina al excepției. Obiectul creat este transmis în e. Prin invocarea e.print(...) este apelată metoda print care tipărește mesajul primit la invocare. Să observăm că în acest mod se evită incrementarea lui na;

• dacă stiva nu este plină, atunci are loc efectiv înscrierea unui nou element în stivă.

Modul în care acționează metoda ia este analog.

Programul este următorul:

Fișierul plina.java :

```
class plina extends Exception {
  void print(String s) {
        System.out.print(s); }
}
```

Fișierul vida.java :

```
class vida extends Exception {
  void print(String s) {
        System.out.print(s); }
}
```

Fișierul s.java :

```
import plina; import vida;
class s {
  int lung, ns; char[] stack;
  s(lung) {
    this.lung=lung ;
    stack = new char[lung];
  }
  void pune(char c) throws plina {
    if (ns==lung) throw new plina();
    else { System.out.print("  P"+c);
           stack[ns]=c; ns++; }
  }
  char ia() throws vida {
    if (ns==0) throw new vida();
    else {
      ns--;
      System.out.print("  I"+stack[ns]);
      return stack[ns];
    }
  }
}
```

Fișierul Stiva.java :

```
import java.util.*;
class Stiva {
  public static void main(String[] sir) {
    int n=10,na,nb,i; char[] a,b; char c;
    a = new char[n]; b = new char[n];
    Random R = new Random();
    for (c='a',i=0; i<n; i++,c++) a[i]=c;
    na=0; nb=0; s Ob = new s(5);
    while (nb!=n) {
      try {
        if (R.nextInt()<0 && na<n) {
            Ob.pune(a[na]); na++; }
        else { c = Ob.ia(); b[nb]=c;
               nb++; }
      }
      catch (vida e) {e.print("  ~I");}
      catch (plina e) {e.print("  ~P");}
    }
    System.out.println();
    for (i=0; i<n; i++)
              System.out.print(b[i]);
    System.out.println();
  }
}
```

Observație. Dacă în clasa Stiva marcăm drept comentarii liniile din interiorul ciclului **while**, cu excepția celor ce încep cu **if** și **else**, atunci nu mai este realizată captarea excepțiilor. Drept urmare, dacă

se ajunge de exemplu la o încercare de a lua un element din stiva vidă, va apare mesajul:

```
vida
  at s.ia (s.java : 13)
  at Stiva.main (Stiva.java : 13)
```

care aduce precizări legate de tipul excepției și de locul unde a intervenit (a fost lansată).

În continuare prezentăm mai exact modul în care funcționează construcția **try**.

Dacă în blocul *bloc* asociat lui **try** nu este lansată nici o excepție, atunci se trece de ultimul **catch**.

Să presupunem acum că din *bloc* este lansată o excepție; fie E tipul său. Se încearcă pe rând, în ordinea în care apar clauzele **catch**, asocierea ei cu una dintre ele. Dacă E coincide cu E_1 sau este derivată din E_1, atunci se execută blocul *bloc_1*, după care se trece de ultima clauză **catch**. În caz contrar se încearcă pe rând asocierea excepției cu E_2,E_3 etc. Dacă nu s-a reușit o asociere, se merge "la un nivel superior" într-o construcție **try** care o cuprinde pe cea în discuție etc. Dacă nici în acest mod nu s-a identificat o asociere, se trece în codul care a invocat metoda curentă etc. În final se va reuși o asociere, deoarece altfel ar apare o eroare chiar în faza de compilare.

Tot o eroare la compilare va apare și dacă de exemplu E_2 este derivată din E_1; explicația constă în faptul că lansarea unei excepții de tipul E_2 nu va putea conduce la captarea ei de către clauza identificată prin E_2 (excepția va fi captată de clauza **catch** precedentă).

Conform celor de mai sus, este posibil ca o construcție **try** să nu aibă nici o clauză **catch**.

Să mai observăm că la lansarea unor excepții din *bloc*, instrucțiunile care urmează în acest bloc nu vor mai fi executate. Mai este clar și că o construcție **try** poate fi utilizată pentru captarea mai multor excepții.

Așa cum am anunțat anterior, o construcție try poate avea și o clauză **finally**. Aceasta permite executarea unei secvențe de instrucțiuni indiferent dacă din blocul *bloc* este lansată sau nu o excepție. Odată inclusă, clauza **finally** nu poate fi în nici un fel ocolită și blocul *bloc_final* este totdeauna executat. Un caz limită este de exemplu următorul:

```
try { ... return 0; ... }
finally { return 1; }
```

pentru care valoarea întoarsă este totdeauna 1.

Prof. dr. Horia Georgescu este profesor la Facultatea de Matematică a Universității din București.

modalitate prezentă în instrucțiunea instr de a ieși din acest ciclu.

În inițializările din **for** este permis să facem declarări, ca de exemplu:

```
for(int i=1;i<=n;i++) s+=i;
```

bineînțeles numai dacă i nu este deja declarat.

Etichete

Înainte de a prezenta alte instrucțiuni, menționăm că o instrucțiune poate fi etichetată. Etichetarea ei se face astfel:

```
eticheta:instructiune
```

unde eticheta este un identificator. În acest mod va fi posibil să facem referire la instrucțiunea respectivă.

Instrucțiunea break

Este folosită pentru a ieși dintr-un bloc. O primă formă este:

```
break;
```

prin care se iese din cel mai interior **for**, **while**, **do** sau **switch** (despre care vom vorbi altă dată) sau bloc ce conține această instrucțiune.

Forma:

```
break eticheta;
```

unde eticheta este eticheta unui bloc sau a uneia dintre instrucțiunile citate, ce conține instrucțiunea **break**. Se poate astfel ieși de exemplu din două instrucțiuni **for** imbricate. Să remarcăm faptul că nu este posibil să trecem controlul oricărei alte instrucțiuni (ca pentru goto din alte limbaje), deci transferul este controlat și respectă structurarea programului.

Instrucțiunea continue

Are forma:

```
continue;
```

și este folosită în cadrul instrucțiunilor **while**, **do** și **for**. Efectul ei este saltul la condiția care controlează reluarea acestor instrucțiuni de ciclare.

Ca și în cazul instrucțiunii **break**, poate fi folosită și varianta etichetată, prin care se poate sări la condiția de control a unei instrucțiuni **while**, **do** sau **for** care o "acoperă" la un nivel superior.

Instrucțiunea return

Termină executarea unei metode și, în cazul când metoda prevede întoarcerea unei valori, furnizează valoarea care va fi întoarsă.

Putem folosi **return** și pentru a ieși dintr-un constructor sau dintr-un bloc de inițializare static.

Câte ceva despre PROCESE

de Cristian Mateescu

Poate v-ați întrebat uneori cum este posibil ca un calculator să execute mai multe programe în paralel... La baza acestui truc stau procesele. În acest episod vom studia facilitățile multitasking și multiuser oferite de Linux precum și câteva utilitare sistem asociate acestora

Conceptul de proces

Ce este un proces? Un proces este un program în execuție. Prin urmare, procesul este dinamic (se modifică în timp) pe când programul este static. Pentru un anume program pot exista la un moment dat unul sau mai multe procese asociate, numite instanțe ale acestui proces.

Fiecare proces are un număr asociat, numit "*pid*" (*process ID* - identificatorul de proces). Aceste identificatoare sunt alocate proceselor în mod crescător, începând cu 0, pe măsura creării acestora. Cu excepția primului proces (0 - creat la inițializarea sistemului), toate celelalte sunt create cu ajutorul funcției sistem **fork**. Programul executabil este încărcat în memorie prin intermediul unei alte funcții sistem, **exec**.

Fiecare proces are acces la un spațiu propriu de adrese, numit spațiu virtual de adrese. Acest spațiu virtual de adrese este cuprins între 0 și adresa virtuală maximă accesibilă respectivului proces. Fiecare astfel de spațiu virtual de adrese este compus din următoarele trei segmente:

- segmentul cod;
- segmentul de date;
- segmentul de stivă.

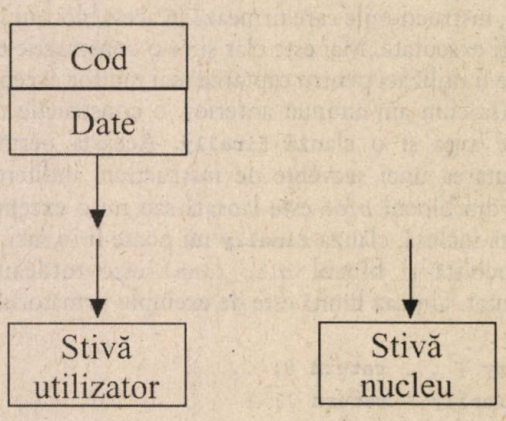
Segmentul de cod conține codul programului aflat în execuție, cel de date conține datele alocate static sau dinamic de procesul respectiv, iar segmentul de stivă conține argumentele, variabilele locale și alte informații necesare funcțiilor procesului respectiv.

Un program se poate executa în două moduri: în mod utilizator (*user mode*) sau în mod nucleu (*kernel mode*). În modul utilizator, fiecare proces nu are acces decât la propria zonă de cod și date precum și la stiva utilizator.

În modul nucleu, programul poate apela instrucțiuni speciale ce îi permit accesul la structurile de date interne ale nucleului.

Fiecare fișier executabil conține un antet, segmentul de text și datele (doar acelea inițializate). Cum procesele se execută atât în mod utilizator cât și în mod nucleu, acestora li se asociază și o stivă nucleu ce va conține informațiile necesare funcțiilor de acces la nucleu. Un proces trece din modul utilizator în modul nucleu când apelează o funcție de sistem sau când este generată o întrerupere.

Segmente și stive



Sensurile de creștere ale segmentelor de date și stivă.

Segmentul de date și stivele utilizator și nucleu sunt dinamice, adică dimensiunea lor variază în timpul execuției (vezi figura "Segmente și stive").

Există câteva funcții mai importante cu ajutorul cărora utilizatorul poate interveni asupra segmentului de date:

funcția **malloc** - alocă o zonă de memorie de dimensiunea cerută;

funcția **calloc** - alocă o zonă de memorie contiguă și o inițializează cu valoarea NULL;

funcția **realloc** - redimensionează o zonă de memorie alocată de una dintre funcțiile **malloc**, **calloc** sau **realloc**;

funcția **free** - eliberează o zonă de memorie alocată de una dintre funcțiile de mai sus.

Atributele proceselor

Fiecare proces are atașate mai multe atribute, dintre care cele mai importante ar fi:

Identificatorul de proces - este un număr întreg mai mare sau egal cu 0, alocat în ordine crescătoare;

Identificatorul de proces al părintelui - este identificatorul de proces al procesului care a lansat funcția **fork** ce a generat procesul respectiv;

ID-ul utilizatorului real;

ID-ul grupului real de utilizatori;

ID-ul utilizatorului efectiv - acestea servesc la determinarea drepturilor de acces la fișiere;

Valoarea *nice* - această valoare este folosită în calculul priorității unui proces: cu cât valoarea sa este mai mare, cu atât prioritatea procesului este mai scăzută.

Listing 1: Lista proceselor

```
[cristi@galaxy cristi]$ ps au
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
cristi     559  0.0  1.5  1496   960   3 S    14:41   0:00 /bin/login -- cristi
cristi     864  0.0  1.5  1492   956   2 S    14:43   0:00 /bin/login -- cristi
cristi     865  0.0  1.2  1184   772   2 S    14:43   0:00 -bash
cristi     886  0.0  1.2  1184   772   3 S    14:45   0:00 -bash
cristi     897  0.1  2.0  2264  1280   3 S    14:45   0:00 lynx
cristi     928  0.0  0.8   860   500   2 R    14:47   0:00 ps au
root       557  0.0  1.5  1496   964   1 S    14:41   0:00 /bin/login -- root
root       560  0.0  0.5   744   324   4 S    14:41   0:00 /sbin/mingetty tty4
root       561  0.0  0.5   744   324   5 S    14:41   0:00 /sbin/mingetty tty5
root       562  0.0  0.5   744   324   6 S    14:41   0:00 /sbin/mingetty tty6
root       585  0.0  1.2  1196   788   1 S    14:42   0:00 -bash
[cristi@galaxy cristi]$
```

Listing 2: Arborele proceselor

```
[cristi@galaxy cristi]$ ps auf
USER       PID %CPU %MEM  SIZE   RSS TTY STAT START   TIME COMMAND
cristi     559  0.0  1.5  1496   960   3 S    14:41   0:00 /bin/login -- cristi
cristi     886  0.0  1.2  1184   772   3 S    14:45   0:00  \_ -bash
cristi     897  0.1  2.0  2264  1280   3 S    14:45   0:00      \_ lynx
cristi     864  0.0  1.5  1492   956   2 S    14:43   0:00 /bin/login -- cristi
cristi     865  0.0  1.2  1184   772   2 S    14:43   0:00  \_ -bash
cristi     930  0.0  0.8   864   520   2 R    14:47   0:00      \_ ps auf
root       557  0.0  1.5  1496   964   1 S    14:41   0:00 /bin/login -- root
root       585  0.0  1.2  1196   788   1 S    14:42   0:00  \_ -bash
root       560  0.0  0.5   744   324   4 S    14:41   0:00 /sbin/mingetty tty4
root       561  0.0  0.5   744   324   5 S    14:41   0:00 /sbin/mingetty tty5
root       562  0.0  0.5   744   324   6 S    14:41   0:00 /sbin/mingetty tty6
[cristi@galaxy cristi]$
```

Stările unui proces

După cum am văzut mai sus, toate procesele cu excepția procesului 0 sunt create prin apelul funcției sistem **fork**. În urma acestui apel, cele trei segmente ale părintelui sunt dublate, ele fiind folosite la inițializarea procesului fiu. În cazul în care există suficientă memorie internă, noul proces va trece în starea "*ready*" (gata pentru execuție), altfel va fi trecut în starea "*ready suspendat*" (va fi swapat pe memoria externă).

Din starea "*ready*" planificatorul de procese (*process scheduler*) îl va trece în starea "*run mod nucleu*", unde activitatea funcției fork se va încheia. Trecerea procesului în această stare se va face pe baza de priorități. Prioritățile sunt dinamice și se calculează în fiecare secundă după formula:

$$\text{prioritate} = \text{baza} + \frac{\text{utilizare\_recenta\_CPU}}{\text{cons tan ta}} + \text{valoare\_m}$$

Pe lângă calculul priorității, rutina ceasului mai are și alte funcții:

- activează periodic procesul *swapper*;
- transmite semnalul SIGALARM către procese;
- măsoară timpul sistem;
- reprogramează ceasul de timp real al sistemului;
- lansează anumite funcții ale nucleului în funcție de anumiți contori interni;
- calculează anumite valori statistice pentru procese.

Când funcția sistem apelată se termină (prima dată aceasta este funcția **fork**), procesul va fi trecut în stare "*run mod utilizator*". Din aceasta stare, procesul poate reveni în starea "*run mod nucleu*" dacă apare o întrerupere sau procesul apelează o funcție sistem.

Când ceasul emite o întrerupere către procesor, procesul care se execută în mod utilizator este trecut în starea "*run mod nucleu*". Când rutina de tratare a întreruperii de ceas își termină activitatea, nucleul poate decide să înceapă execuția unui alt proces. În acest caz, procesul curent va fi trecut în starea "*ready preemptat*". În această stare ajung procesele care sunt executate în mod nucleu și trebuie să revină în mod utilizator, dar este planificat pentru execuție un alt proces mai prioritar. Un proces care se află în starea *preemptat* poate fi *swappat*. Când un proces ce se află în starea "*ready preemptat*" este planificat pentru execuție, el este trecut direct în starea "*run mod utilizator*".

Întreruperile sunt tratate ca procese privilegiate și, prin urmare, rulează în mod nucleu. Când un proces ce se află în execuție în mod nucleu (deci a apelat o funcție sistem) trebuie să aștepte un eveniment, nucleul lansează funcția **sleep** și îl trece în starea "*așteptare*". Procesele pot fi în așteptare pentru evenimente cu probabilitate mare de a se produce (eliberare de buffere, de i-noduri, terminarea unei operații cu discul) dar pot fi și în așteptarea unor evenimente pentru care există probabilitatea de a nu se produce. Din acest motiv, funcția **sleep** este apelată de către nucleu împreună cu un parametru care precizează prioritatea.

În funcție de valoarea acestei priorități, procesul respectiv va fi sau nu va fi sensibil la primirea unui semnal din partea nucleului. Astfel, nucleul are posibilitatea de a controla procesele prin trimiterea unor semnale către acestea. Când respectivul eveniment s-a produs, toate evenimentele din coadă legate de el vor fi trecute în starea "*ready*" prin funcția sistem **wakeup**.

Pentru terminare, un proces apelează funcția **exit**. Prin urmare, acesta va trece în starea "*run mod nucleu*" iar apoi va trece în starea *terminat* ("*zombie*"). În aceasta stare el își păstrează intrarea din tabela proceselor, însă nu mai dispune de nici o resursă. Această intrare va fi eliberată prin intervenția părintelui acestui proces sau a procesului *init*.

Utilitare

În continuare, voi prezenta pe scurt câteva utilitare sistem ce sunt folosite pentru gestiunea proceselor în Linux. Cel mai simplu utilitar utilizat pentru a obține informații despre procesele ce rulează la un moment dat în sistem este *ps*.

```
[cristi@galaxy cristi]$ ps
  PID TTY STAT TIME COMMAND
  559   3 S    0:00 /bin/login -- cristi
  864   2 S    0:00 /bin/login -- cristi
  865   2 S    0:00 -bash
  886   3 S    0:00 -bash
  897   3 S    0:00 lynx
  924   2 R    0:00 ps
[cristi@galaxy cristi]$
```

Comanda *ps* acceptă diverse argumente. În modul în care a fost apelat în primul caz (fără nici un fel de argumente), *ps* listează doar procesele utilizatorului curent. Dacă dorim să listăm toate procesele care rulează la un momente dat în sistem, vom folosi o comanda de genul "*ps au*" (vezi **Listing 1**)

Rolul argumentului "*u*" este să listeze procesele tuturor utilizatorilor din sistem, iar cel al lui "*a*" este de a afișa toate informațiile despre aceste procese. Putem afișa, de asemenea, un arbore al proceselor, în care se poate urmări care este părintele fiecărui proces:

```
[cristi@galaxy cristi]$ ps f
  PID TTY STAT TIME COMMAND
  559   3 S    0:00 /bin/login -- cristi
  886   3 S    0:00  \_ -bash
  897   3 S    0:00      \_ lynx
  864   2 S    0:00 /bin/login -- cristi
  865   2 S    0:00  \_ -bash
  925   2 R    0:00      \_ ps f
[cristi@galaxy cristi]$
```

Aceste argumente se pot folosi și combinate, astfel că putem afișa sub formă de arbore procesele tuturor utilizatorilor din sistem (vezi **Listing 2**).

Un alt utilitar important, cu ajutorul căruia putem termina execuția unor procese, este "*kill*". Acesta primește ca argument *pid*-ul procesului pe care dorim să îl terminăm. Să urmărim exemplul de mai jos:

```
[cristi@galaxy cristi]$ ps
  PID TTY STAT TIME COMMAND
  559   3 S    0:00 /bin/login -- cristi
  864   2 S    0:00 /bin/login -- cristi
  865   2 S    0:00 -bash
  886   3 S    0:00 -bash
  897   3 S    0:00 lynx
  924   2 R    0:00 ps
[cristi@galaxy cristi]$ kill 897
[cristi@galaxy cristi]$ ps
  PID TTY STAT TIME COMMAND
  559   3 S    0:00 /bin/login -- cristi
  864   2 S    0:00 /bin/login -- cristi
  865   2 S    0:00 -bash
  886   3 S    0:00 -bash
  924   2 R    0:00 ps
[cristi@galaxy cristi]$
```

După cum se poate observa în rândurile ce urmează instrucțiunii "*kill*", procesul terminat nu mai apare.

Un alt utilitar cu ajutorul căruia se pot obține informații despre procesele ce rulează pe sistem este "*top*". Acesta este un utilitar *full-screen* ce afișează în timp real informații despre procese. În **Listing 3** puteți vedea cum arată un ecran al utilitarului "*top*".

Cristian Mateescu poate fi contactat prin intermediul redacției la adresa mateescu@agora.ro sau la adresa cristim@fx.ro.

Listing 3: Utilitarul top

```
[cristi@galaxy cristi]$ top

2:48pm  up 7 min,  3 users,  load average: 0.00, 0.04, 0.01
59 processes: 58 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  1.8% user,  2.8% system,  0.0% nice, 95.3% idle
Mem:    62372K av,  28132K used,  34240K free,  43832K shrd,   1412K buff
Swap:   48156K av,      0K used,  48156K free                   9136K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
  937 cristi    19   0   728  728   560 R       0  4.7  1.1   0:00 top
    1 root      0   0   420  420   352 S       0  0.0  0.6   0:02 init
    2 root      0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root      0   0     0    0     0 SW      0  0.0  0.0   0:00 kswapd
    4 root      0   0     0    0     0 SW      0  0.0  0.0   0:00 md_thread
   82 root      0   0   368  368   316 S       0  0.0  0.5   0:00 kerneld
   93 root      0   0   404  404   352 S       0  0.0  0.6   0:00 apmd
  230 root      0   0   464  464   384 S       0  0.0  0.7   0:00 syslogd
  239 root      0   0   432  432   312 S       0  0.0  0.6   0:00 klogd
  251 daemon    0   0   420  420   344 S       0  0.0  0.6   0:00 atd
  262 root      0   0   488  488   408 S       0  0.0  0.7   0:00 crond
  273 bin       0   0   356  356   288 S       0  0.0  0.5   0:00 portmap
  284 root      0   0   776  776   452 S       0  0.0  1.2   0:00 snmpd
  295 root      0   0   424  424   348 S       0  0.0  0.6   0:00 inetd
  306 root      0   0   636  636   516 S       0  0.0  1.0   0:00 named
  317 root      0   0   440  440   364 S       0  0.0  0.7   0:00 routed
  328 root      0   0   348  348   312 S       0  0.0  0.5   0:00 lpd
  338 root      6   0   404  404   348 S       0  0.0  0.6   0:00 lpd

[cristi@galaxy cristi]$
```

Simplu, dar nu simplist

OBERON:
Magia componentelor

Mircea Sârbu

Limbajul pe care vi l-am prezentat în primăvară, Logo, a fost unul destul de diferit de cele studiate de regulă în școlile din țară. În schimb, Oberon-2 este o variantă de Pascal care vi se va părea familiară de la bun început. Cu toate acestea, Oberon este mai mult decât un limbaj. Este o filozofie a programării.

Inițial am încercat să realizez pentru GInfo o prezentare a limbajului și mediului Oberon pe baza unui articol pe care l-am scris cu vreo doi ani în urmă pentru revista BYTE România. Am lucrat o vreme pe această linie, dar... *A nice try* — cum ar zice americanii. Mi-am dat seama că problemele pe care încearcă să le rezolve Oberon nu prea vă sunt familiare, iar noțiunilor de sisteme de operare și ingineria programării implicate de o astfel de abordare nu se numără printre cele studiate în licee — uneori nici chiar în facultățile de specialitate, din păcate...

Așadar, în acest număr vă voi prezenta doar o introducere succintă (dar absolut necesară), care are rolul de a pune în evidență conceptul central pe care se bazează Oberon: extensibilitatea. Partea de programare se vrea o exemplificare a simplității și puterii limbajului.

Necazuri în Paradis

Este un fapt incontestabil că producția de software se află actualmente într-o criză profundă. Cauzele acestei crize sunt multiple, dar pot fi împărțite — într-un mod destul de grosier — în două categorii.

Prima categorie se referă la evoluțiile tehnologice din informatica modernă, care au indus un nivel de complexitate situat peste posibilitățile instrumentarului software actual. Răspândirea calculatoarelor personale și descentralizarea procesării pe care rețele de PC-uri au adus-o reprezintă poate cele mai importante astfel de evoluții. În acest sens, se poate aminti impunerea interfețelor grafice, care a obligat proiectanții de software să-și schimbe radical modul de concepere și construcție a aplicațiilor. De asemenea, modelul client-server a bulversat profund obișnuințele dobândite în era procesării centralizate realizate de marile computere (*mainframe*).

A doua categorie de cauze se referă la lipsa de maturitate a producției de software, care nu a reușit încă să adopte metode industriale, rămânând deocamdată cantonată într-o zonă aparținând mai degrabă artizanatului. În acest sens este demn de menționat întârzierea adoptării (sau impunerii) unor standarde larg acceptate, care să permită *asamblarea* aplicațiilor din componente interoperabile realizate de terți producători.

Ar mai fi, desigur, și diverse alte motive și s-ar putea discuta despre relațiile dintre cauzele dintre cele două categorii (de pildă faptul că lipsa standardelor este o consecință a evoluției extrem de rapide a domeniului). Cert este însă că cererea imensă de software și graba producătorilor de a o satisface au condus la cumulări necontrolate de cod, rezultând tot mai multe produse nesigure, insuficient testate, pline de "scame" și mari consumatoare de resurse. Este tot mai evident că programe însumând milioane de linii sursă sunt — la modul practic — imposibil de controlat, testat și depanat în mod eficient.

Extensibilitate

Altfel spus, lucrurile au devenit prea complicate. Oberon reprezintă o tentativă de a simplifica cât mai mult dezvoltarea de software prin susținerea consecventă a unui set de principii clare.

Ideea centrală care stă la baza edificiului este extensibilitatea. Aceasta înseamnă că orice "program" trebuie să ofere o funcționalitate de bază, care să poată fi apoi extinsă (sau adaptată) în funcție de nevoile specifice ale utilizatorului. Pentru atingerea acestui deziderat, cercetătorii de la ETH, în frunte cu Niklaus Wirth, au recurs la două instrumente strâns corelate: un limbaj orientat pe componente și un mediu de lucru extensibil (*framework*).

Ambele sunt desemnate prin același nume: Oberon.

Limbajul Oberon este un descendent al limbajelor Pascal (de la care a moștenit simplitatea și expresivitatea) și Modula-2 (de la care a moștenit puterea și modularitatea). Aparent paradoxal, Oberon este mai simplu decât Pascal și mai puternic decât Modula-2. Principala noutate pe care Oberon o aduce este orientarea pe obiecte, asupra căruia mă voi concentra într-un articol special.

Mediul de lucru Oberon reprezintă cadrul care permite dezvoltarea și rularea aplicațiilor scrise în Oberon. De fapt, acest cadru constă dintr-o ierarhie de module dezvoltate în limbajul Oberon. Deoarece noțiunea de "*framework*" este pentru majoritatea cititorilor nouă, voi insista ceva mai mult asupra ei în rândurile ce urmează.

Framework

De regulă, proiectantul de software se folosește de un mediu de dezvoltare, care cuprinde în mod uzual un editor de cod sursă, un compilator, un depanator (*debugger*) interactiv, un editor de legături (*linker*) și niște instrumente CASE (*Computer Aided Software Engineering*). Dintre acestea din urmă, au devenit locuri comune instrumentele vizuale pentru construirea interfeței și, tot mai adesea, instrumente pentru modelarea structurilor de date. Pentru realizarea aplicațiilor, proiectantul folosește adesea biblioteci (realizate "în casă" sau cumpărate) cuprinzând subrutine sau, în cazul limbajelor orientate pe obiecte, ierarhii de clase. Proiectanții produc cu ajutorul acestui instrumentar aplicații finite, cu funcționalitate bine definită și, eventual, biblioteci de rutine statice (utilizabile la realizarea altor programe) sau dinamice (utilizate la execuția altor programe).

Mediul de lucru Oberon cuprinde și el un subsistem de dezvoltare, compus în principiu din aceleași elemente. În plus însă, Oberon mai cuprinde un nucleu (*core*), un subsistem de text și un subsistem de forme. Ultimele două subsisteme permit lucrul cu documente compuse, fiind folosite și în cadrul subsistemului de dezvoltare, pentru editarea codului sursă și a interfeței (dar nu numai).

Nucleul Oberon cuprinde un executiv (*run-time*) pentru module Oberon, un încărcător (*loader*) cu verificarea versiunilor, un colector de memorie reziduală (*garbage-collector*), o bibliotecă de clase sub forma unei colecții de module Oberon.

Prezența unui executiv poate să pară stranie și, cu siguranță, va duce cu gândul pe mulți cititori la ideea că este vorba de o interpretare de cod. Nu este așa: codul realizat de compilator este cod nativ pe 32 de biți. Explicația prezenței acestui nucleu este cheia extensibilității întregului sistem.

Un *framework*, în cazul de față Oberon, poate fi considerat o aplicație nefinalizată (sau, mai bine zis, nesfârșită). El oferă un cadru general cuprinzând multe elemente de uz general (cum ar fi subsistemul de text), dar care poate fi îmbogățit și specializat cu noi componente. Ceea ce este esențial este faptul că nu voi fi nevoit să

Scurtă istorie

În întreaga sa carieră, profesorul Niklaus Wirth s-a condus după un celebru dicton al lui Einstein: *Make it as simple as possible, but not simpler* ("fă-o cât mai simplu cu putință, dar nu mai simplu decât atât"). Așa se explică faptul că limbajul Pascal, pe care l-a creat în anul 1970 la ETH Zurich, a avut un succes deosebit, fiind încă și astăzi unul dintre limbajele cele mai folosite în educație.

Pe la mijlocul anilor '70, Wirth pornește un alt proiect extrem de ambițios: dezvoltare unui nou computer (*desktop workstation*) care să fie complet programabil într-un limbaj de nivel înalt. Acest proiect a condus la realizarea limbajului Modula-2 (specificațiile au fost publicate în 1979). Acesta a reținut multe dintre trăsăturile Pascalului, la care s-a adăugat modularitatea și posibilitatea de evitare controlată a sistemului de tipuri a limbajului, pentru a permite și programarea low-level (de pildă pentru drivere de periferice). Astfel, întregul sistem de operare a putut fi scris în Modula-2.

Între timp, "incubatorul de idei" de la Xerox PARC (*Palo Alto Research Center*) au adus în scenă elemente noi, care au sporit interesul lui Wirth pentru computerele desktop (să nu uităm că la acea vreme PC-urile încă nu apăruseră). Aici a fost inventat mouse-ul și imprimanta cu laser, aici s-a născut interfața grafică și noțiunea de orientare pe obiecte. Tot aici a fost dezvoltat limbajul Smalltalk, care, alături de orientare pe obiecte, introduce noțiunea de *framework* și utilizează o interfață grafică, servind totodată și ca sistem de operare.

Pornind de la aceste idei, este dezvoltat un nou sistem de operare, numit *Cedar*, care se baza însă pe un limbaj asemănător Pascalului. Rezultatul a fost impresionant, dar prea complicat pentru gusturile lui Wirth.

În 1984, apare o nouă provocare: la ETH este lansat proiectul unei noi stații de lucru, numite *Ceres*. Wirth, împreună cu Jürg Gutknecht se decid să rafineze și să simplifice sistemul Cedar, astfel încât să construiască un sistem extensibil, bazat pe componente care să poată fi încărcate în mod dinamic, atunci când sunt solicitate. Pentru aceasta însă au nevoie de un limbaj de operare care să susțină unele caracteristici specifice programării orientate pe obiecte, cum ar fi încapsularea informației (*information hiding*), legarea târzie (*late binding*) și polimorfismul.

Pentru a face față acestor cerințe, Wirth simplifică până la esență limbajul Pascal, preia din Modula-2 modularitatea (care implică încapsularea) și variabilele de tip procedural (care reprezintă o formă de legare la momentul execuției). Pentru a obține și polimorfismul (evitând însă terminologia obiectuală), Wirth adaugă posibilitatea de *extensie* a tipului *record*. Astfel, oriunde este acceptat un anumit tip, sunt acceptate și extensiile acestuia.

Astfel a luat naștere limbajul Oberon. Sistemul de operare (care s-a numit la fel) a constat din doar 12.000 de linii de cod și rula în 2 MB de memorie!

Versiuni

Limbajul Oberon a suferit o revizie substanțială în 1992, când Wirth împreună cu colegul său de la ETH, profesorul Mössenböck, au adus câteva îmbunătățiri, dintre care se distinge introducerea procedurilor tipizate (*type-bound procedures*). Versiunea revizuită s-a numit Oberon-2 și reprezintă actualmente standardul *de-facto* al limbajului.

La ETH, proiectul Oberon a fost continuat de generații de studenți. Au fost dezvoltate pentru stații Ceres aplicații dintre cele mai diverse, de la birotică până la CAD (*Computer Aided Design*). Răspândirea limbajului și a sistemului de operare era însă limitată de răspândirea stațiilor Ceres. Aceasta a condus la portarea sistemului pe diverse tipuri de calculatoare, inclusiv calculatoare personale, cum sunt Macintosh și cele compatibile IBM PC. Toate versiunile dezvolate la ETH au fost și sunt gratuite.

Versiunea actuală cea mai răspândită este *Oberon System 3*, întreținută de ETH. System 3 nu mai este un sistem de operare propriu zis, ci se instalează *peste* anumite sisteme de operare: Windows și Linux pentru PC-uri Intel, și respectiv MacOS pentru mașini Macintosh. Pentru mașini Intel este disponibilă și o versiune care se instalează ca sistem de operare (cunoscută sub numele *PC Native Oberon*).

Sistemul poate fi descărcat (împreună cu întreg codul sursă) de la adresa:
ftp://ftp.inf.ethz.ch/pub/Oberon/System3/

O altă versiune disponibilă la ETH este *Oberon V4*, care este se fapt sistemul Oberon original cu o interfață textuală. Disponibilă prin ftp la adresa:
ftp://ftp.inf.ethz.ch/pub/software/Oberon/OberonV4/

Toate versiunile dezvoltate la ETH se bazează pe interfața originală dezvoltată pentru mașinile Ceres. Utilizarea unui mouse cu trei butoane și sistemul propriu de ferestre (care nu se suprapun) nu este însă în concordanță cu actualele standarde din lumea comercială (unde domină sistemul Windows). Acest fapt a limitat utilizarea sistemului în aplicații comercială, cu toate că este folosit în numeroase universități pentru studiu.

Pentru a stimula răspândirea tehnologiilor revoluționare de la ETH în mediile comerciale, în 1993 a fost înființată firma *Oberon microsystems, Inc.*, avându-l pe profesorul Wirth în consiliul de conducere. Firma a dezvoltat o nouă versiune a mediului Oberon care, spre deosebire de cele de la ETH, folosește interfața grafică și sistemul de fișiere a sistemului de operare a mașinii gazdă. Au fost dezvoltate două versiuni: una pentru Windows și una pentru Macintosh, perfect compatibile la nivelul codului sursă. Inițial, mediul s-a numit Oberon/F iar limbajul Oberon/L. În anul 1997, Oberon microsystems adaugă câteva extensii limbajului Oberon-2 și îl denumește ***Component Pascal***. Mediul de lucru a fost la rândul său re-botezat, numindu-se acum ***BlackBox Component Builder***. O versiune educațională (gratuită) este disponibilă la adresa www.oberon.ch/prod/BlackBox/download.html

Articolul de față se referă la Component Pascal și BlackBox Component Builder, versiunea 1.3. Pentru simplitate, am folosit doar denumirea consacrată: **Oberon**.

recompilez întregul ansamblu pentru a obține interactivitatea cu noua componentă. Este suficient să obțin codul executabil, care va fi încărcat și lansat în execuție atunci când aplicația va avea nevoie să utilizeze componenta.

Pentru a ilustra diferența dintre un *framework* și o simplă bibliotecă de rutine, să considerăm cazul unui editor de text. Un text este în general definit ca o secvență de elemente, care sunt de regulă caractere (însoțite de o serie de atribute, cum ar fi fontul, dimensiunea, stilul, culoarea etc.) și elemente de formatare (de pildă caractere de control). Definiția tipurilor de date și implementarea operațiilor ce pot fi executate se concretizează de obicei într-o colecție de rutine.

Dacă aceste rutine formează o bibliotecă statică (de pildă o colecție de unit-uri Pascal), atunci editorul va cuprinde întreaga bibliotecă și se va concretizează într-un program executabil, al cărui cod este inaccesibil utilizatorului. Dacă rutinele sunt plasate într-o bibliotecă dinamică (de pildă DLL — *Dynamic Link Library*), atunci aceasta va fi livrată împreună cu programul executabil. În ambele cazuri, funcționalitatea editorului nu poate fi modificată decât de producător, care trebuie să modifice codul sursă și să recompileze întreg ansamblul.

Subsistemului de text din Oberon este la rândul său compus dintr-un set de module care definesc și implementează tipul text, împreună cu operațiile de bază asupra acestuia. Diferența este că acest subsistem face parte din mediul de lucru și poate fi extins de către utilizator, deoarece interfețele modulelor sunt publice iar extinderea nu implică nici un fel de recompilare. Dacă funcționalitatea intrinsecă a subsistemului nu mă mulțumește, am posibilitatea să programez în Oberon comenzi noi, pe baza celor existente deja în subsistem. Pentru ca acestea să devină funcționale nu trebuie decât să le compilez și să le apelez (de exemplu le pot include în meniuri). Ele vor fi dinamic încărcate de către *loader* și vor conlucra cu celelalte componente ale mediului.

Componente software

Spre deosebire de cazul bibliotecii de rutine, în care codul nou (al aplicației) apelează codul vechi (subprogramele din bibliotecă, de pildă), în cazul unui astfel de mediu de lucru apare și situația în care codul vechi (în cazul de față *framework*-ul) apelează codul nou. De pildă subsistemul de text, parcurgând caracterele unui document, întâlnește un element care nu este caracter (de pildă este o tabelă de calcul), situație în care trebuie să apeleze componenta corespunzătoare. Siguranța sistemului în ansamblu nu poate fi asigurată prin mijloacele proprii ale unui limbaj de programare, deoarece nu se poate face nici o supoziție despre modul în care proiectantul respectivei componente a implementat, de pildă, gestionarea memoriei. Deoarece ideea componentelor software se bazează tocmai pe legarea târzie (*late binding*) a componentelor (adică la momentul execuției), este limpede că un mediu bazat pe componente trebuie să se bazeze pe un *run-time* care să

sprijine execuția în siguranță a codului.

Executivul verifică, de pildă, compatibilitatea tipurilor pentru a evita utilizarea incorectă a interfeței între module. Un apel incorect conduce la o eroare locală, care nu afectează restul sistemului. Dacă, de exemplu, un document conține un element de un tip necunoscut sistemului, el este considerat un alien și este ignorat. Cu toate acestea el va supraviețui operațiunilor de copiere, salvare, restaurare, etc. De îndată ce codul componentei corespunzătoare este adăugat sistemului, elementul respectiv "prinde viață".

Abstractizare

În general, limbajele de programare și mediile de lucru sunt plasate pe o axă teoretică, pe care ne-o putem imagina ca fiind verticală. Direcțiile acestei axe corespund nivelului de abstractizare. Astfel, un limbaj sau un mediu de lucru care face uz de particularitățile concrete ale mașinii se numește "de nivel jos" (*low level*), în timp ce un limbaj sau mediu de lucru care lucrează cu abstracții (noțiuni generale, care pot fi particularizate pentru orice suport hardware sau software) este considerat "de nivel înalt" (sau *high level*).

Într-o asemenea abordare, un limbaj cum este Pascal ar putea fi considerat de nivel mediu. Limbajul C ar putea fi plasat undeva mai jos, deoarece este mai apropiat de sistemul de operare (care este, prin definiție, de nivel jos). Mergând în jos pe axă, vom întâlni limbajele de asamblare și, foarte jos, limbajele-mașină corespunzătoare diferitelor procesoare.

Mergând pe axă în sus, vom întâlni, de pildă, limbajele de acces la baze de date (cum ar fi SQL — considerat "de nivel foarte înalt") și mediile de dezvoltare care fac uz de aceste limbaje (numite adesea 4GL — limbaje de generația a patra).

Limbajul Oberon are, în acest context, o situație cu totul specială. Practic, el poate fi considerat în egală măsură de nivel foarte jos (s-au dezvoltat sisteme de operare în Oberon!) și de nivel foarte înalt (poate fi utilizat într-o manieră similară cu limbajele 4GL). Dar mediul de lucru Oberon este, cu siguranță, de nivel foarte înalt, aspect asupra căruia voi insista în rândurile care urmează.

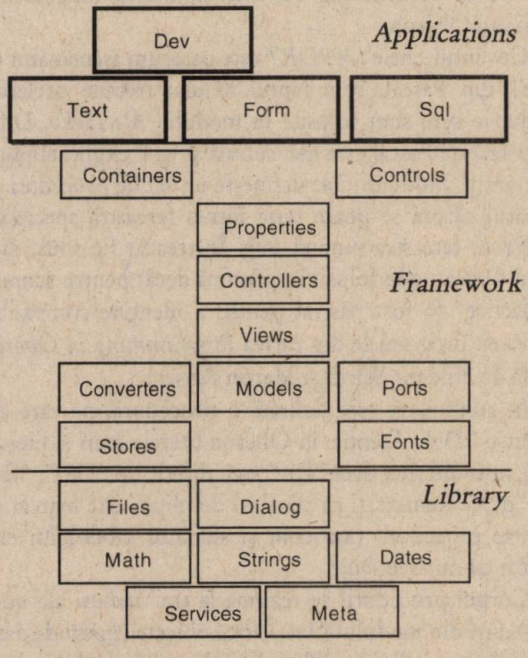
Oberon, ca *framework*, poate fi considerat o extensie a sistemului de operare. Dacă sistemul de operare abstractizează detaliile particulare ale pilotării dispozitivelor periferice sau ale interacțiuni cu utilizatorul, Oberon abstractizează și mai mult ceea ce oferă sistemul de operare. De pildă, în Oberon nu avem nevoie de fișiere și nici de ferestre, ambele fiind cuprinse implicit în noțiunea de *document*. Dezvoltarea întregului mediu (realizată — să nu uităm — în limbajul Oberon) a fost astfel concepută încât să nu facă nici un fel de supoziție privind particularitățile de nivel jos. De pildă, nu contează dacă sistemul de fișiere al sistemului gazdă este sau nu arborescent. De asemenea, interfața grafică este implicit cea furnizată de sistemul de operare, fără însă ca programatorul să aibă de-a face cu particularitățile acesteia. Rezultatul este că documentele sunt portabile la nivel binar iar aplicațiile sunt portabile la nivel sursă. O aplicație dezvoltată în Oberon pe Windows va rula în mediul Oberon pe Macintosh, preluând automat interfață specifică a acestuia.

Ce urmează

S-ar putea ca această scurtă introducere să pară multor cititori confuză, deoarece amestecă două noțiuni aparent distincte: un limbaj de programare și un mediu de lucru. Spun "aparent", deoarece există o strânsă dependență între limbaj și mediu. Dar aceast strânsă dependență va deveni vizibilă abia după ce vom conoaște câteva aspecte ale programării. Deci... Toate la vremea lor. O incursiune în limbaj va lămuri unele aspecte și, sper, vă va trezi gustul pentru Oberon. În funcție de ecoul acestui articol, poate vom continua prezentarea unor aspecte interesante.

Ierarhia de module

Un dreptunghi mic semnifică un modul. Un modul care importă un alt modul este plasat mai sus în ierarhie. Un dreptunghi mare reprezintă un întreg subsistem, constând din mai multe module.
Ierarhia este structurată în trei straturi:
– **Biblioteca** (Library) - conține module care furnizează servicii independente;
– **Mediul** (Framework) - furnizează drivere de dispozitive precum și abstacțiunile de bază pentru obiectele persistente, pentru obiectele editabile și pentru ierarhii de obiecte editabile, adică documente.
– **Aplicații** (Application) - furnizează trei subsisteme standard: un subsistem de text, un subsistem de forme (ex. casete de dialog) și un subsistem de dezvoltare.
Diagrama prezintă doar modulele cele mai importante.



Programare în Oberon

Pentru oricine cunoaște limbajul Pascal, Oberon va părea familiar. Așa că, fără prea multe pregătiri, putem trece la prezentarea unui prim program.

Hello world

În mod tradițional, primul program într-un nou limbaj de programare trebuie să afișeze pe ecran cuvintele "Hello world". Să respectăm deci tradiția:

```
MODULE MirHello;

    IMPORT Out;

    PROCEDURE Do*;
    BEGIN
            Out.Ln;
            Out.String("Hello world");
            Out.Ln
    END Do;

END MirHello.
```

Simplu, nu-i așa? Într-adevăr, dar...

În primul rând, spre deosebire de Pascal, nu apare cuvântul cheie *PROGRAM*, ci *MODULE*. Explicația este faptul că în Oberon unitatea de compilare este modulul. De altfel, în Oberon nu există noțiunea de "program", deoarece un program este ceva finit, bine determinat, ceea ce vine în contradicție cu extensibilitatea intrinsecă a mediului Oberon.

Cuvântul-cheie *IMPORT* este oarecum asemănător cu *USES* din Pascal, prin faptul că aici trebuie declarate modulele care sunt folosite în modulul *MirHello*. Diferența față de Pascal este mai subtilă și va fi explicată puțin mai târziu. Modulul *Out* definește un set de proceduri cu ajutorul cărora se poate scrie într-o fereastră specială a mediului, fereastră numită *Log*. În treacăt fie spus, modulul *Out* nu este folosit în practică decât pentru scopuri "didactice" (a fost păstrat pentru a menține compatibilitatea cu programele din cartea *Programming in Oberon* scrisă de Niklaus Wirth și Martin Reiser).

În continuare este definită o procedură, pe care am numit-o "*Do*". Atenție: în Oberon literele mari și literele mici sunt diferite, deci "Do" este diferit de "DO", "do" sau "dO". Remarcați că sfârșitul definiției este marcat cu numele procedurii (așa cum și sfârșitul modulului este marcat cu numele său).

Corpul procedurii se rezumă la trei apeluri ale unor proceduri din modulul *Out*. Orice obiecte (tipuri de date, variabile, proceduri etc.) definite într-un alt modul trebuie prefixate cu numele modulului respectiv. Procedura *Out.Ln* nu face altceva decât să ducă cursorul la începutul liniei următoare din *Log*, iar *Out.String* va scrie în *Log* șirul de caractere pe care-l primește ca parametru. Șirul de caractere poate fi încadrat fie între ghilimele (ca în acest exemplu), fie între apostrofuri (ca în Pascal). Dacă șirul cuprinde ghilimele, atunci încadrarea se face cu apostrofuri și invers.

În fine, să mai remarcăm că nu există în acest caz ceva analog cu "programul principal" din Pascal.

Am lăsat anume la sfârșit explicația steluței (asterisc) care urmează numelui procedurii nou definite. Ea semnifică faptul că procedura în cauză (*Do*) este *exportată* de modulul *MirHello*. Aceasta înseamnă că ea este vizibilă și accesibilă din orice modul care importă modulul *MirHello*. Ajungem astfel la noțiunea de *interfață*, extrem de importantă în Oberon.

Orice modul în Oberon dispune de o interfață, care constă din toate simbolurile exportate de modul. Interfața este publică și este singura cale de a accesa funcționalitatea unui modul. Interfața unui modul nu este doar o noțiune teoretică, ci ea se concretizează într-un fișier de simboluri, pe care compilatorul îl creează odată cu codul executabil, pe baza sursei modulului și a fișierelor de simboluri ale modulelor importate. Fișierele de simboluri cuprind și informațiile semantice legate de simbolurile definite, astfel încât compilatorul poate verifica consistența utilizării lor.

În fine, cum putem folosi modulul pe care l-am scris? Există două posibilități: fie să fie importat de către un alt modul, fie interactiv. Cea mai simplă variantă interactivă este să scriem într-un document (orice apare într-o fereastră a mediului Oberon este un document — deci poate fi editat) șirul de caractere "MirHello.Do", sa-l selectăm (exact ca în orice editor de text) și apoi să lansăm (din meniul *Dev*) comanda *Execute*. Mediul Oberon va încărca în memorie codul executabil al modulului (dacă nu era deja încărcat) și va lansa în execuție procedura. Descărcarea modulului se poate face doar explicit (prin comanda *Unload* din meniul *Dev*).

Calculul numerelor prime

Următorul exemplu încearcă să pună în evidență unele diferențe față de Pascal și, totodată, să demonstreze cât de ușor se poate construi o interfață grafică simplă. Vom porni de la un program Pascal care calculează numerele prime mai mici decât o un anumit număr întreg furnizat de utilizator (Notă: am preluat programul Pascal dintr-un articol scris de Tiberiu Socaciu. Precizez că programul nu urmărește performanța). În caseta "Numere prime" sunt prezentate, alăturat, sursele Pascal și Oberon.

Numere prime în Pascal și Oberon

```
program Prime;

uses crt;

var
  i, n, p: integer;
  x: array [1 .. 1000] of integer;
       {sirul numerelor prime}

function prim (i: integer): boolean;
  var
    j, k: integer;
  begin
    j := 2;
    k := 0; {indicator de primalitate}
    while j < i do
      begin
      if i mod j = 0 then
        {aici am gasit un divizor j pentru i}
        k := 1;
      j := j + 1
      end;
    prim := k = 0
  end;

begin
  clrscr;
  write ('n=');
  readln (n);
  {p = numarul de numere prime}
  p := 0;
  for i := 2 to n do
    begin
    if prim (i) then
      begin
      p := p + 1;
      x [p] := i
      end
    end;
  {afisam cele p numere prime}
  for i := 1 to p do
    write (x [i], ' ')
end.
```

```
MODULE MirPrime;
  IMPORT Out;
  TYPE Tablou = POINTER TO ARRAY OF INTEGER;
  VAR
    i, n, p: INTEGER;
    x: Tablou;  (* Sirul numerelor prime *)
    primeDlg*: RECORD
      lim*: INTEGER;
    END;
  PROCEDURE Prim (i: INTEGER): BOOLEAN;
  VAR j, k: INTEGER;
  BEGIN
    j := 2; k := 0; (* indicator de primalitate *)
    WHILE j < i DO
      IF i MOD j = 0 THEN (* am gasit un divizor *)
        k := 1
      END;
      j := j + 1
    END;
    RETURN k = 0
  END Prim;

  PROCEDURE Do*;
  BEGIN
    n := primeDlg.lim;
    NEW(x, n);
    p := 0; (*p = numarul de numere prime*)
    FOR i := 2 TO n DO
      IF Prim(i) THEN
        x[p] := i; p := p +1
      END
    END;
    FOR i :=  0 TO p-1 DO
      Out.Int(x[i], 0);
      Out.Ln;
    END
  END Do;

BEGIN
  Out.String("Modulul MirPrime a fost incarcat");
  Out.Ln
CLOSE
  Out.String("Modulul MirPrime a fost descarcat");
  Out.Ln
END MirPrime.
```

Un prim șoc este declarația tipului *Tablou*. De ce nu sunt precizate limitele? De ce este pointer? Să le luăm pe rând...

În primul rând, în Oberon indicele unui tablou nu poate fi decât de tip întreg. Mai mult, s-a renunțat la precizarea limitelor. Se poate preciza doar numărul elementelor. De pildă, dacă am fi urmărit programul Pascal, declarația s-ar fi scris astfel:

x : ARRAY 1000 OF INTEGER;

Valorile legale ale unui indice în tabloul *x* sunt cuprinse între 0 și 999.

Dar, în Oberon există așa-numitele tablouri deschise

(*open arrays*), tablouri care se alocă doar la momentul execuției. Fiind o structură dinamică, utilizarea ei tipică este cea din exemplu (ca pointer). Alocarea se poate face (prin procedura standard *NEW*) când deja se cunoaște numărul maxim al elementelor, rezultând astfel o substanțială economie de memorie și un spor de siguranță (nu există riscul depășirii numărului de elemente specificat la compilare).

Ați remarcat, desigur, că declararea unui pointer nu se mai face prin construcția (destul de criptică) "^Tip" ci printr-una mai clară: *POINTER TO Tip*.

Este cazul să remarcăm forma modificată a comentariilor: *(* text *)*. Menționez în treacăt că în Oberon comentariile pot fi imbricate. Ați încercat vreodată în Pascal să scoateți în comentariu o întreagă procedură și să constatați că sunteți "taxat" de compilator pentru că in procedură existau deja comentarii? Dacă da, veți simți diferența...

Nu vom insista acum asupra variabilei *primeDlg*, a cărei utilizare se leagă de interfața grafică a aplicației. Vom reveni asupra ei la momentul potrivit.

Ajungem la funcția *Prim* și constatăm că este declarată tot prin cuvântul cheie *PROCEDURE*. Într-adevăr, în Oberon nu există declarația *FUNCTION*. O funcție nu este altceva decât o procedură care returnează o valoare (în cazul de față de tip boolean). Să remarcăm că valoare este returnată explicit, prin instrucțiunea *RETURN* (spre deosebire de Pascal, unde se returnează printr-o variabilă implicită).

Urmărind corpul funcției, descoperim câteva modificări sintactice. Instrucțiunile *WHILE* și *IF* se termină întotdeauna prin *END*, eliminând astfel obligativitatea încadrării secvențelor de instrucțiuni între *BEGIN* și *END*. Același lucru este valabil și pentru alte instrucțiuni (*LOOP, FOR, WITH*).

Deși nu reiese din exemplul acesta, mă simt dator să precizez că instrucțiunea *IF* beneficiază de o sintaxă extinsă, care permite evitarea problemelor legate de *dangling else* (vezi articolul pe această temă din acest număr al revistei). În afară de varianta cu *ELSE*, există și o variantă cu *ELSIF*, care permite înlănțuirea testelor, în felul următor:

```
IF cond1 THEN instr1
ELSIF cond2 THEN instr2
ELSE instr3 END
```

Înainte de a trece la procedura *Do*, să remarcăm că funcția *Prim* nu este exportată, ceea ce înseamnă că ea nu poate fi apelată decât din modulul *MirPrime*.

Procedura *Do* este în schimb exportată, ceea ce va permite apelarea ei din cadrul unui dialog. Dar, cum despre aceasta vom discuta mai târziu trecem și peste prima instrucțiune, care are doar rolul de a prelua de la utilizator valoarea *n*.

Observăm că de îndată ce dispunem de această valoare putem aloca tabloul dinamic, prin procedura standard *NEW* (deoarece *x* este un pointer).

Așa cum am precizat deja, indicele tabloului *x* pornește de la zero (și nu de la 1), ceea ce conduce la ușoare ajustări (întâi se face atribuirea și abia apoi incrementarea).

În fine, o ultimă observație legată de procedura *Do* se referă la dereferențierea implicită a pointerilor. Deoarece Oberon a exclus aritmetica pointerilor (considerată foarte riscantă), o variabilă de tip pointer se referă la variabila pe care o indică (operatorul de dereferențiere ^poate fi omis).

Și... surpriză! Apare "programul principal". De fapt nu este așa. Secvențele de instrucțiuni introduse prin BEGIN și CLOSE sunt executate doar atunci când modulul este încărcat în memorie și respectiv descărcat din memorie. Utilitatea acestor secvențe nu rezultă din exemplul prezentat, dar este bine să fiți informați despre încărcarea și descărcarea modulelor, doarece o nouă compilare nu încarcă noul cod în memorie, așa că riscați să rulați vechiul cod și să vă mirați de ce nu-și modifică atitudinea...

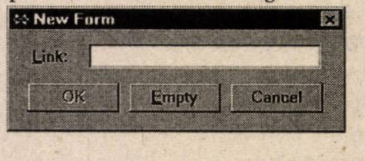
Interfața cu utilizatorul

În fine, a venit timpul să ne ocupăm și de felul în care putem cere în mod elegant utilizatorului să furnizeze valoarea până la care să se calculeze numerele prime.

În Oberon, subsistemul de forme permite construirea foarte ușoară a casetelor de dialog, a formularelor de culegere a datelor și a altor elemente ale interfeței grafice cu utilizatorul.

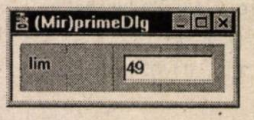
Voi descrie pas cu pas crearea dialogului.

Pasul 1: Din meniul *Controls* selectăm comanda *New Form*. Va apărea o fereastră de dialog:



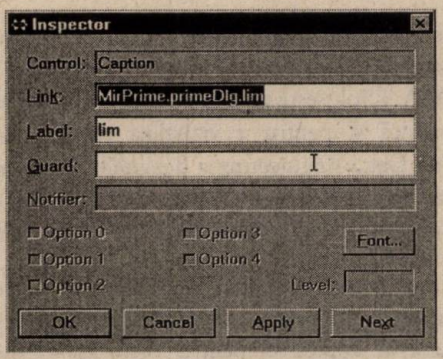
Ni se cere deci să legăm noua formă de o variabilă. Vom furniza numele variabilei *MirPrime.primeDlg*. Să notăm că acesta este motivul pentru care am exportat această variabilă.

Pasul 2. Subsistemul de forme va încerca să ne ajute și va furniza o primă formă a dialogului:

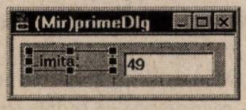


Un început modest, dar promițător. Restul muncii trebuie să o facem manual:

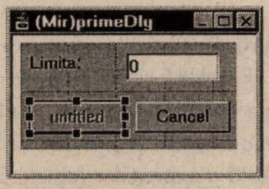
a) Mofificăm mesajul (*Caption*) afișat înaintea câmpului. În loc de "lim" vom scrie "Limita:". Pentru aceasta selectăm obiectul de pe ecran (de tip *Caption*) și, făcând clic pe butonul din dreapta al mouse-lui, se desfășoară un mic meniu, din care vom alege comanda *Properties*. Vom fi puși în față unei noi ferestre de dialog:



Tot ce avem de făcut este să modific în mod corespunzător câmpul *Label*. După ce scriem aici "Limita:" și apasam pe butonul *OK*, dialogul nostru va arăta așa:

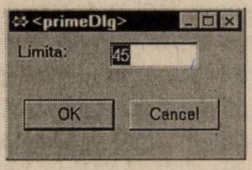


b) Următorul pas va fi să-i adugăm două butoane. Pentru aceasta va trebui mai întâi să-l mărim (tragem cu mouse-ul de colțul din dreapta-jos). Apoi adăugăm un buton de comandă (din meniul *Controls*) și un buton *Cancel* (din același meniu). Le așezăm sub câmpul pentru introducerea limitei și lansăm comanda *Recalc Focus Size* (din meniul *Layout*). După câteva ajustări, obținem ceva de genul următor:



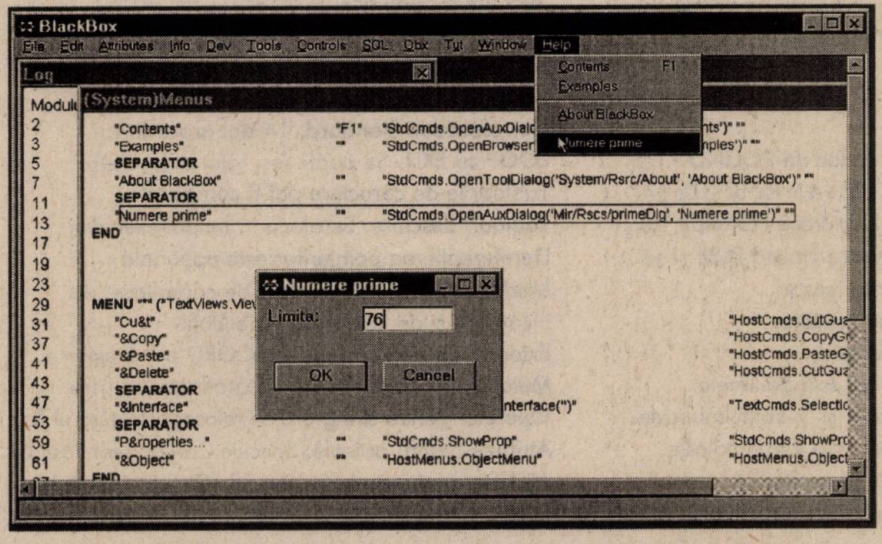
c) Cum butonul de comandă este selectat, bănuiesc că v-ați prins că vom lansa din nou inspectorul. Singura treabă ce-o avem de facut este să completăm câmpul *Link* cu numele procedurii care va fi lasată când va fi apăsat butonul (desigur, "MirPrime.Do"), să specificăm în câmpul *Label* textul ce va fi scris pe buton (să zicem "Start") și să bifăm căsuța *Default*, indicând astfel că acesta va fi butonul implicit selectat. Rezultatul obținut îl salvăm într-un anumit subdirector al directorului BlackBox (să zicem ..\Mir\Rscs).

d) Putem să testăm în această fază funcționarea dialogului, prin comanda *Open as Aux Dialog* din meniul *Controls*. Va fi afișat dialogul în forma sa funcțională (până acum se afla în faza de proiectare):



Dacă vom introduce limita 45 și vom apăsa pe *OK*, în fereastra *Log* vor curge toate numerele prime mai mici decât 45.

Pasul 4. Nu ne mai rămâne acum decât să introducem într-un meniu o comandă care să deschidă dialogul pe care tocmai l-am creat, urmând ca acesta la rândul lui să apeleze procedura *Do* a modulului *MirPrime*. Pentru aceasta va trebui să deschidem un document care cuprinde specificația meniurilor (o putem face simplu, prin comanda *Menus* din meniul *Info*). Vom insera în meniul *Help* încă un separator, urmat de o linie în care vom preciza textul noii opțiuni din meniu (să zicem "Numere prime"), urmat eventual de un *short-cut* (nu e cazul, scriem ""), apoi de comanda care va fi executată și numele ferestrei de dialog (*"StdCmds.OpenAuxDialog('Mir/Rscs/primeDlg', 'Numere prime')"*), după care ar trebui să urmeze un *guard* (o procedură care să verifice dacă sunt îndeplinite condițiile pentru a lansa comanda). Deoarece nu avem nici un guard, scriem șirul vid (""). În fine, actualizăm sistemul de meniuri (prin comanda *Update menus* din același meniu *Info*) și... gata!



În captura alăturată se poate vedea opțiunea nou introdusă în meniul *Help*, fereastra *(System)Menus* în care s-a făcut specificarea acestei noi opțiuni (rândul evidențiat) și dialogul pe care l-am creat (*Numere prime*). În partea din stânga se poate vedea și un fragment din fereastra *Log*, în care sunt înscrise rezultatele ultimei rulări a comenzii *MirPrime.Do*.

Promisiuni

Chiar și din exemplele simple prezentate până acum rezultă cu claritate — cred eu — că în Oberon se poate face tot ce se poate face în Pascal, și chiar mai mult decât atât. Posibilitatea de a crea în mod foarte simplu interfețe elegante cu utilizatorul nu este la îndemâna multor medii de dezvoltare.

Dar adevărata putere a limbajului și a mediului Oberon abia de aici începe. Programarea obiectuală și extensibilitatea permit construirea unor componente oricât de complexe. Versiunea comercială pentru mediul Windows dezvoltată de firma Oberon microsystems este dotată cu un subsistem de interfațare cu baze de date, cu un subsistem de comunicații TCP/IP și poate conlucra cu alte aplicații Windows (de pildă cu Word sau Excel).

Există deja multe aplicații reale bazate pe Oberon. Un medic folosește o aplicație realizată în Oberon pentru achiziția și procesarea datelor furnizate de echipamentele de laborator iar o editură și-a dezvoltat o aplicație specifică de machetare pentru un ziar. Societatea de asigurări Schweizer Ruck folosește BlackBox Component Builder pentru instruirea inginerilor software. Siemens AG utilizează același mediu ca front-end pentru baze de date, iar NASA îl folosește pentru a rezolva probleme de termodinamică și pentru a vizualiza rezultatele. Firma elvețiană Delta Engineering a dezvoltat o aplicație profesională pentru proiectarea etichetelor cu coduri de bare, iar diverse universități dezvoltă la rândul lor aplicații cu Component Pascal.

În fine, o tehnologie bazată pe Oberon dezvoltată la University of California at Irvine, numită *Juice*, permite realizarea de applet-uri pentru Web, asemănătoare cu cele dezvoltate în Java, însă mult mai rapide.

Chiar dacă Oberon nu va deveni niciodată un succes comercial, faptul că ideile pe care se bazează au influențat sau vor influența tehnologiile noi din lumea informaticii este suficient de important. Într-un recent interviu acordat revistei Wired, Billy Joy - unul dintre creatorii limbajului Java - recunoștea că în laboratorul echipei sale funcționează o stație care rulează Oberon...

Mircea Sârbu este directorul editorial al editurii Computer Press Agora. Poate fi contacta prin e-mail la adresa msarbu@agora.ro.

Diferențe între Pascal și Oberon-2

Caracteristici eliminate:

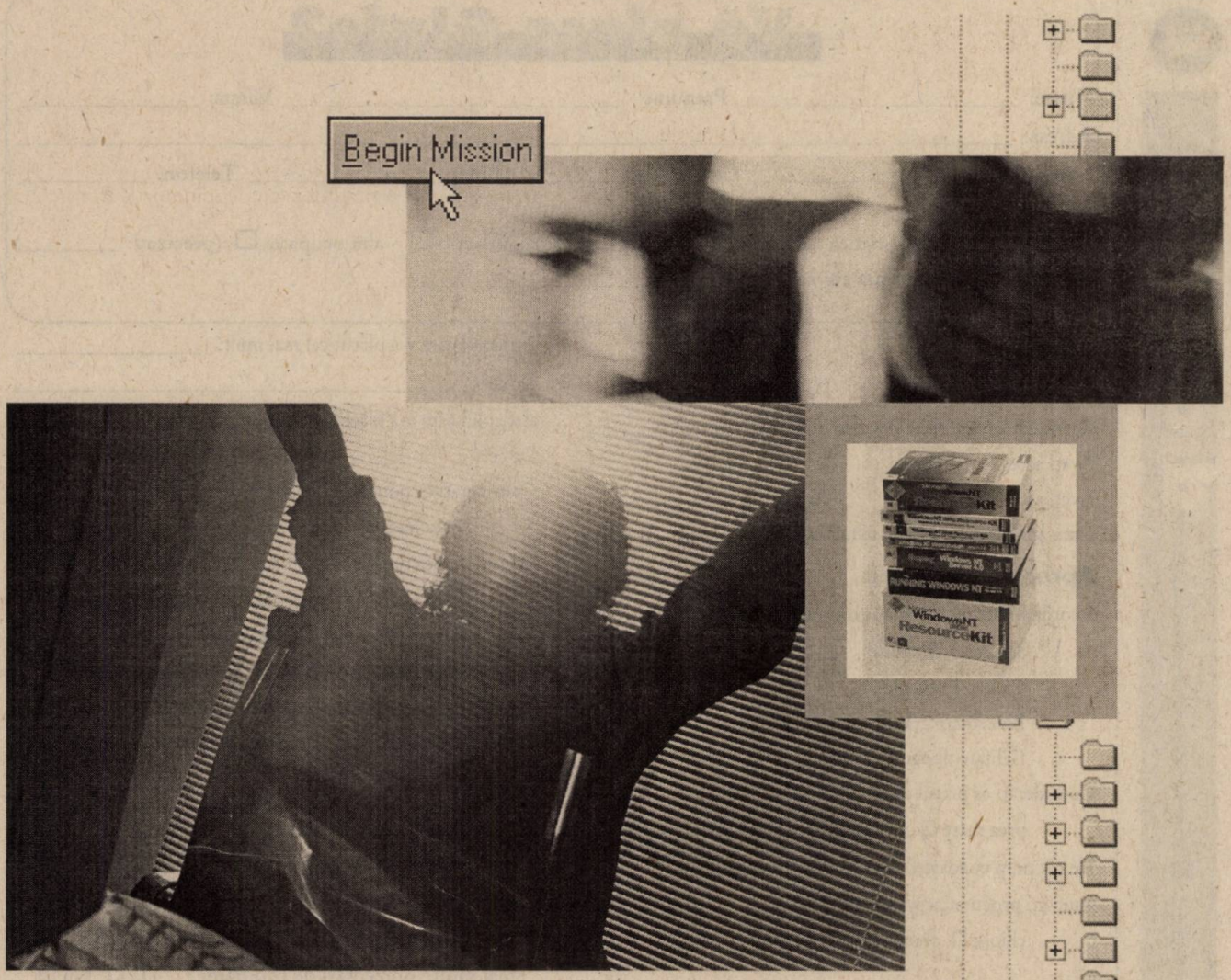
Tipuri subdomeniu. Se folosesc tipuri întregi standard.
Tipuri enumerate. Se folosesc constante întregi.
Indecși de tip oarecare. Indecși sunt definți pe domeniul întreg 0..max-1.
Nu există mulțimi generale. Tipul SET poate conține întregi de la 0 la 31
Nu există DISPOSE explicit. Memoria eliberată este colectată automat.
Nu există articole cu variante. Se folosește extensia.
Nu exista GOTO.
Nu exista funcțiile standard PRED și SUCC. Se folosesc funcțiile INC și DEC, definite pe întregi.
Nu există facilități de I/O predefinite.Se folosesc rutine de bibliotecă.

Caracteristici modificate:

Procedura standard ENTIER în loc de ROUND.
Sintaxa constantelor reale. 3.0E+4 în loc de 3.0e+4.
Sintaxa declarării pointerilor. Declarația POINTER TO.
Sintaxa instrucțiunii CASE. Apare clauza ELSE și se folosește "|" ca separator între cazuri.
Numele procedurilor trebuie repetate.
Literele mari și mici sunt diferite.
Sintaxa stringurilor. Apostrofuri sau ghilimele.
Comentarii. Se marchează cu (* și *). Pot fi imbricate.
Paranteze pentru mulțimi. Se scriu între acolade.
Sintaxa funcțiilor. Se declară ca proceduri. Returnează valoori prin instrucțiunea RETURN.
Declarații. CONST, TYPE și VAR pot apăra în orice ordine.
Tipul procedură. Pot fi declarate și variabile.
END explicit. IF, WHILE și LOOP se termină cu END.
ELSIF. IF poate avea mai multe ramuri.
BY în FOR. Înlocuiește TO și DOWNTO.
Evaluarea scurtă a expresiilor logice. Evaluarea se termină de îndată ce se determină rezultatul.
Constante declarate prin expresii.
Conversii explicite cu SHORT.

Caracteristici noi

Numere și caractere hexa. Ex: 100H (= 256)
Noi tipuri numerice. LONGINT, SHORTINT, BYTE, SHORTREAL.
Scăderea mulțimilor. Diferența simetrică este definită.
Noi proceduri standard. 14 noi proceduri.
LOOP cu EXIT. Se poate ieși fotat din LOOP.
Tablourile de caractere pot fi comparate.
Tablouri deschise. Se alocă în timpul execuției
Dereferențierea pointerilor este opțională.
Module. Reprezintă unuitate de compilare, de încărcare și de limitare a vizibilității.
Extensia tipurilor. Tipurile RECORD pot fi extinse.
Metode. Procedurile pot fi atașate unui tip.
Operator pentru string-uri. Se folosește selectorul S.
Atribute pentru articole. Specific Component Pascal.
Atribute pentru metode. Specific Component Pascal.



Microsoft Press helps

you

create your own answers

WHERE DO you want to go today? ® Microsoft Press

COMPUTER PRESS AGORA

COMPUTER PRESS AGORA

Sucursala București, Str. Poterași 35/2, Tel.01-3309281, Fax: 01-3309285

Vă place GInfo?

Nume: _____ Prenume: _____ Vârsta: _____

Adresa: _____

Localitate: _____ Cod poștal: _____ Județ: _____ Telefon: _____

Ocupația: elev informatică ☐ elev ☐ student ☐ cadru didactic ☐ altă ocupație ☐ (precizați: _____)

Școala (firma, organizația etc.): _____

Dotări

Folosiți un calculator acasă? Da ☐ Nu ☐

Câte calculatoare sunt instalate în școală: (aprox.) _____

Aveți acces la e-mail: Da ☐ Nu ☐

Aveți acces la WWW: Da ☐ Nu ☐

Cum apreciați dotarea cu tehnică de calcul a școlii: _____

Aprecieri generale

Acordați note de la 1 la 10 pe următoarele criterii:

Structura tematică: ___

Calitatea articolelor: ___

Prezentare grafică: ___

Calitate tipografică: ___

Considerați că prețul este:

prea mare ☐ accesibil ☐ prea mic ☐

De cât timp cunoașteți Gazeta de Informatică (ani)? _____

Cum ați procurat acest număr:

chioșc ☐ abonament ☐ cadou ☐ altfel ☐

În acest număr

Apreciați cu note de la 1 la 10 articolele acestui număr:

Indexare cu B-arbori ___

Programe autverificabile ___

Despre dangling-else ___

Unix - scurtă istorie ___

Excepții în Java ___

Linux: Procese ___

Motoare de căutare ___

Oberon: magia componentelor ___

Articolul care v-a plăcut cel mai mult: _____

De ce? _____

Articolul care v-a plăcut cel mai puțin: _____

De ce? _____

Considerați că ponderea problemelor rezolvate este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea problemelor propuse este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea subiectelor teoretice este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că nivelul de accesibilitate al articolelor este

prea cobotât ☐ potrivit ☐ prea ridicat ☐

În viitor

În următoarele numere așteptați:

mai multe articole de programare? Da ☐ Nu ☐

mai multe articole teoretice? Da ☐ Nu ☐

mai multe articole despre Internet? Da ☐ Nu ☐

mai multe prezentări de programe? Da ☐ Nu ☐

prezentări de jocuri? Da ☐ Nu ☐

articole despre hardware? Da ☐ Nu ☐

V-ar plăcea să citiți:

pagini redactate de elevii unor licee? Da ☐ Nu ☐

lucrări premiate, realizate de elevi? Da ☐ Nu ☐

actualități școlare? Da ☐ Nu ☐

prezentarea unor situri Web? Da ☐ Nu ☐

tips & tricks? Da ☐ Nu ☐

Comentarii:

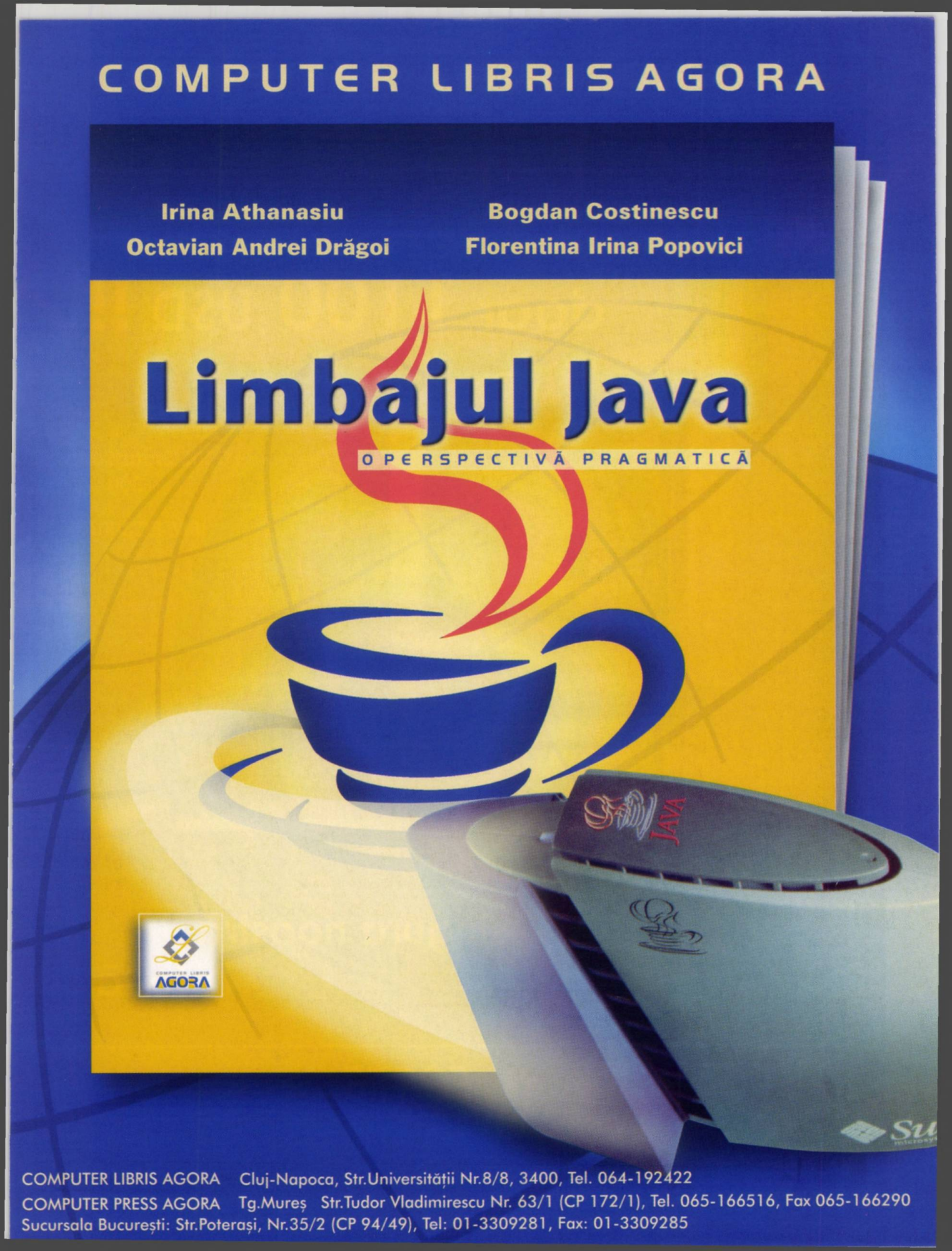
Dacă nu vă este suficient spațiul, folosiți coli albe suplimentare

COMPUTER LIBRIS AGORA

Irina Athanasiu
Octavian Andrei Drăgoi
Bogdan Costinescu
Florentina Irina Popovici

Limbajul Java

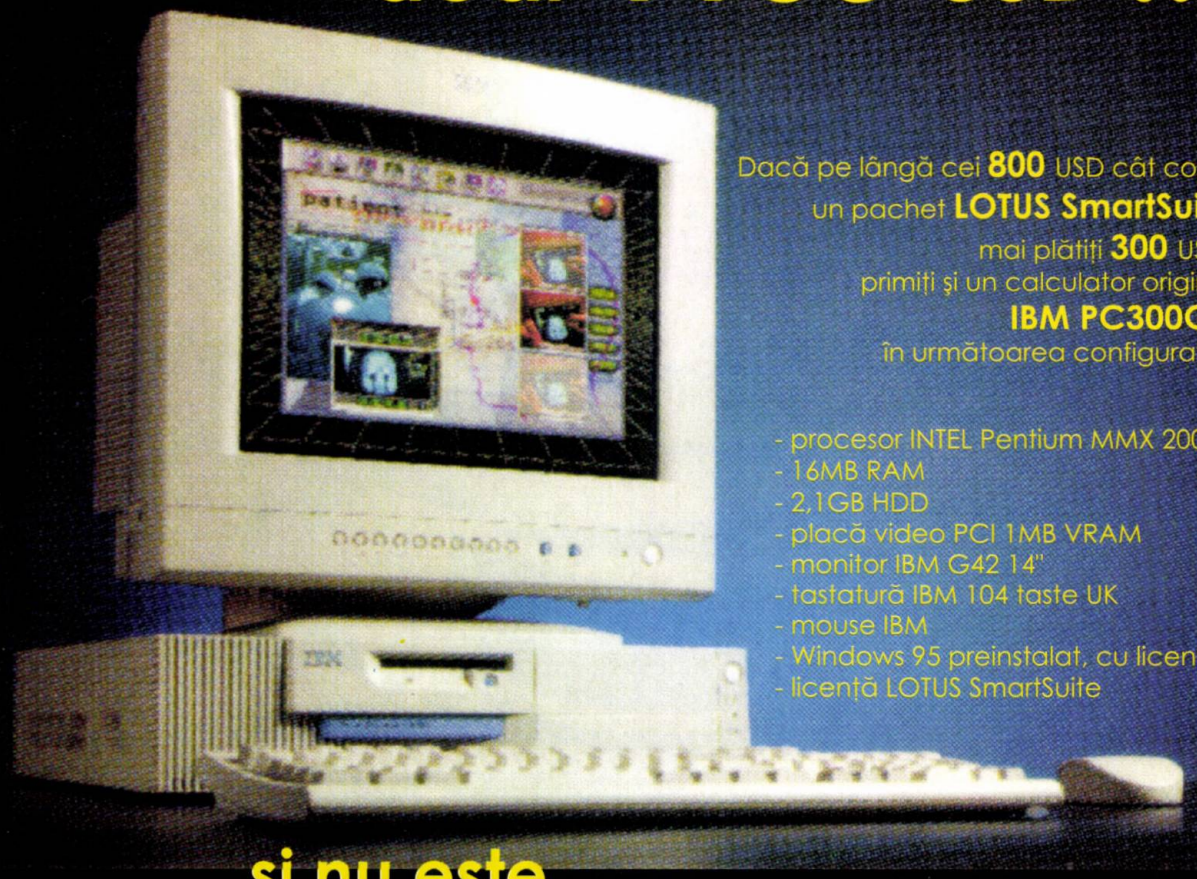
O PERSPECTIVĂ PRAGMATICĂ



COMPUTER LIBRIS AGORA Cluj-Napoca, Str.Universității Nr.8/8, 3400, Tel. 064-192422
COMPUTER PRESS AGORA Tg.Mureș Str.Tudor Vladimirescu Nr. 63/1 (CP 172/1), Tel. 065-166516, Fax 065-166290
Sucursala București: Str.Poterași, Nr.35/2 (CP 94/49), Tel: 01-3309281, Fax: 01-3309285

Ştiţi cât costă un IBM PC?

doar 1100 USD !!!



Dacă pe lângă cei **800** USD cât costă un pachet **LOTUS SmartSuite** mai plătiţi **300** USD, primiţi şi un calculator original **IBM PC300GL**, în următoarea configuraţie:

- procesor INTEL Pentium MMX 200MHz
- 16MB RAM
- 2,1GB HDD
- placă video PCI 1MB VRAM
- monitor IBM G42 14"
- tastatură IBM 104 taste UK
- mouse IBM
- Windows 95 preinstalat, cu licenţă
- licenţă LOTUS SmartSuite

şi nu este singura noastră ofertă !!!

IBM® Business Partner

Microsoft
WHERE DO YOU WANT TO GO TODAY ?™
Authorized Distribution Partner/
DSP for Romania

FORTE COMPUTERS
Flexibility.Made in Romania

IBM® Business Partner

Lipscani 102, Bucureşti
Tel: 3122360/6137282/3120948 Fax: 3122630

Octombrie 1998
Vol. 8/7

Lei 8000

gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

ARBORI STEINER
PROBLEMA FLUXULUI MAXIM

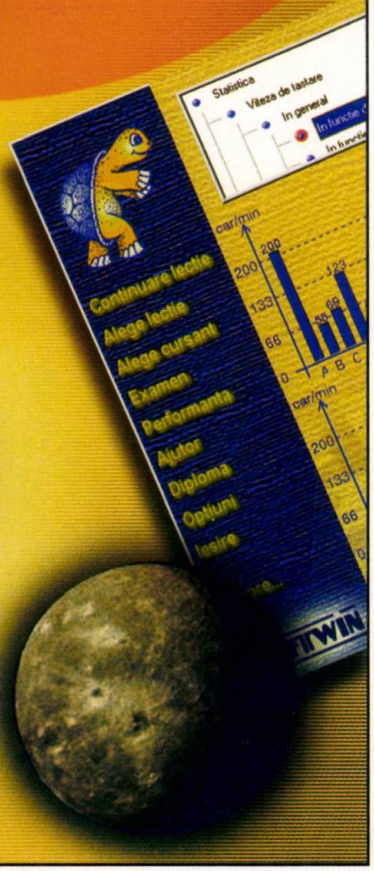
FOCUS

TEHNO
DELPHI

ACTUAL
Software educațional

SERIAL
Pagini WEB cu HTML 3.2

BABEL
Oberon



PLUS
Internet
Interviu
SOLUTII

Expeditor
s.c."Computer Press Agora"s.r.l.
C.P. 172 O.P. 1
RO - 4300 Târgu-Mureș
T.P.
Aprobat R.A.P.R.
Nr.103/DC/053/1998
Valabil - 1998 -
INFADRES

COMPUTER PRESS AGORA

ÎNTR-O LUME ÎN PERPETUĂ SCHIMBARE,

ÎN CARE LUNAR SUNT LANSATE PE PIAȚĂ

MII DE TEHNOLOGII ȘI PRODUSE NOI,

AI NEVOIE DE O **CĂLĂUZĂ !**

PC MAGAZINE ROMÂNIA

te ajută oferindu-ți tot ceea ce-ți este necesar pentru a te orienta printre produsele și tehnologiile informatice de ultimă oră. PC Magazine România îți oferă articole de prezentare și evaluare, teste de laborator și analize comparative, opinii ale unor experți recunoscuți și îți prezintă într-o manieră accesibilă cele mai moderne tehnologii.

Citește **PC MAGAZINE ROMÂNIA** și vei găsi

CALEA SPRE CEA MAI BUNĂ ALEGERE

COMPUTER PRESS AGORA, Str.Tudor Vladimirescu Nr.63/1, 4300 Tg.Mureș, Tel. 065-166516, Fax 065-1662190

Vol. 8, Nr. 7 1998

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu* (București)
Redactor șef:
prof. *Clara Ionescu* (Cluj)
Editor executiv:
Mircea Sârbu (Tg-Mureș)
Au mai colaborat:
Ioan Cozac (Târgu-Mureș)
Mihai Scorțaru (Cluj-Napoca)
Horia Giurgiu (Cluj-Napoca)
Cristian Matei (Târgu-Mureș)
Eleonor Ciurea (Brașov)
Drăgoș Marian (Târgu-Mureș)
Dan Iancu (București)

Tehnoredactare:
Călin Borz, Dan Crețu
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
C.P. 172-1, 4300 Tg-Mureș
Adresa e-mail: office@ginfo.ro
Tel: (065) 166.516
Fax: (065) 166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *S.C. Infopress srl*,
Odorheiu Secuiesc,
tel: 066.216.483
Editor: Computer Press Agora srl
str. T. Vladimirescu nr. 63/1
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director editorial:
Mircea Sârbu
Director tehnic:
Szabó László
AGORA
Vânzare reclamă:
Adrian Alexandru (*aalex@agora.ro*)
Mihaela Szabo (*mszabo@agora.ro*)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: (01) 330.92.81

sumar

soluții.. PAG. 5

probleme REZOLVATE
PAGINI REALIZATE DE Clara IONESCU
Algoritmi și implementări propuse de autorii problemelor

probleme.. PAG. 15

propuse pentru REZOLVARE
***
Lotul olimpic s-a antrenat la Constanța cu aceste probleme

interviu.. PAG. 19

Școala, viitorul și informatica
Interviu cu dl. *Horia Georgescu*

actual.. PAG. 22

Software educațional
Cu două prezentări: RoDactilo și Windows '95 Tutorial

babel.. PAG. 52

Oberon: Obiecte și tipuri
Mircea SÂRBU
Mai mult decât un limbaj: o filosofie.

focus.. PAG. 25

Arbori Steiner
Horia GIURGIU

Problema fluxului maxim
Eleonor CIUREA

tehno.. PAG. 31

Object Pascal și Delphi
Dragoș MARIAN

Indexare cu "hărți de biți"
Ion COZAC

serial.. PAG. 39

Intrări/ieșiri și compilare în JAVA
Horia GEORGESCU

Pagini WEB cu HTML 3.2
Mihai SCORȚARU

internet.. PAG. 49

Motoare de căutare
Dan IANCU

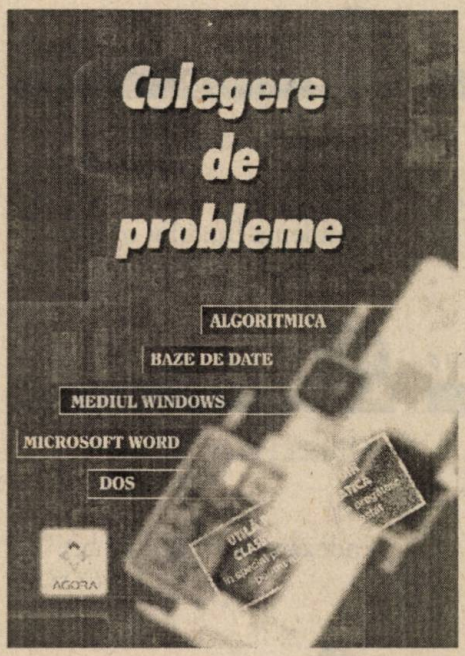
dialog.. PAG. 64

Poșta redacției

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Culegerea conține:

- probleme de algoritmică, majoritatea rezolvate în Pascal
- probleme de gestiune a bazelor de date (FOX-PRO)
- probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word
- probleme de verificare a cunoștințelor aferente sistemului de operare DOS

Această culegere constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru atestatul de informatică dar poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, culegerea conține o varietate mare de enunțuri de probleme rezolvate, abordând diferite metode și stiluri de programare.

• • •

Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) mai elegant și cu o eficiență în programare mai ridicată.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422] sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

INCREDIBIL...

dar ADEVĂRAT !

S-a întâmplat în Portugalia, la cea de a 10-a ediție a Olimpiadei Internaționale de Informatică. România a fost solicitată oficial să organizeze în 2007 cea de a 19-a ediție a acestei olimpiade. Este o recunoaștere a învățământului românesc de informatică? Va fi o încercare grea? Oare trebuie să așteptăm anul 2007 să ne lămurim în privința forțele noastre proprii?

A X-a ediție a IOI

Între 5-12 septembrie, anul curent, orașul Setúbal din Portugalia a găzduit cea mai așteptată și, totodată, cea mai grea competiție de informatică organizată la nivel mondial pentru tineri între 15 și 19 ani.

Echipa României s-a pregătit temeinic de-a lungul mai multor luni (ani...) pentru a-și demonstra în fața întregii lumi inteligența, perseverența, forța și, evident, cunoștințele de nivel înalt. Nu cred că munca lor nu valorează nimic și că ar fi o întâmplare sau un "moft" să participi la olimpiade. "Trebuie să te compari cu absolutul", m-a sfătuit nu o dată profesorul meu de matematică... "trebuie să cauți să te autodepășești, nu doar să te compari cu alții". La rândul meu, așa îi învăț și eu pe cei care tind să se mulțumească cu "rezultate parțiale".

Dar să ne întoarcem la IOI. Rezultatele au fost frumoase, toți cei patru concurenți români au venit acasă cu medalii: *Mihai Stroe*, București - aur absolut (punctaj maxim), *Tudor Leu*, Focșani - argint, *Mihai Scorțaru*, Cluj - argint și *Diodor Bițan*, București - bronz. Cu aceste rezultate, România s-a calificat pe locul 6 din 61 de țări participante.

Nu pot continua aceste gânduri adresând doar felicitări concurenților. În spatele succesului lor se ascund mult prea multe ore de muncă, mult efort - evident în primul rând din partea lor, dar și din partea profesorilor și părinților. Și nu doar atât. A fost nevoie de talent și de ambiție. A fost nevoie de preocupare din partea Ministerului Educației. Nu pot să nu amintesc sponsorii, fără al căror sprijin multe competiții pregătitoare nu ar fi fost posibile.

Nu vreau să continuu enumerarea. Evident, nu este vorba de un rezultat de excepție; încă de la primele participări la IOI , echipa României s-a evidențiat printre echipele cu cele mai bune rezultate.

Trecutul

Prima ediție IOI a fost organizată în orașul Pravetz din Bulgaria, în mai 1989. Au participat 13 țări. La această ediție România nu a participat, chiar dacă ea "s-a petrecut" în vecini.

A doua ediție (1990) s-a desfășurat în Bielorusia, în orașul Minsk, cu participarea a 25 de echipe. Concurenții români au revenit în țară cu o medalie de argint și trei mențiuni. Dar a fost prima participare și toți membrii echipei României au fost evidențiați. În Grecia, la Atena, "luptându-se" cu 26 echipe, băieții din România au obținut trei medalii de bronz. În 1992 a avut loc o creștere explozivă. Succesul olimpiadelor internaționale de informatică a atras la Bonn, în Germania, 51 de țări participante (scorul României: un argint și un bronz).

În 1993, a cincea ediție a adus ceva extraordinar: la Mendoza, în Argentina, România a câștigat locul I pe echipe (două medalii de aur, un argint și un bronz), *Radu Lupșa* obținând aur absolut. Atenția lumii s-a întors cu interes spre România, spre învățământul nostru de informatică. Acest rezultat nu a fost deocamdată egalat și va rămâne în istoria olimpiadelor ca unul de excepție.

A urmat, în 1994, Haninge (Suedia), unde am obținut, într-o competiție puternică, un aur, două medalii de argint și un bronz. Apoi, în Olanda (la Eindhoven), un aur și trei medalii de argint. Ediția a 8-a s-a organizat din nou într-o țară vecină, în Ungaria, la Veszprém. Rezultatele României au strălucit din nou, dar de data aceasta cele patru medalii de argint au mai spus ceva: echipa României era omogenă, diferența de punctaj între primul și al patrulea fiind foarte mică. În

1997, Africa de Sud a prins "pe picior greșit" pe mulți dintre concurenți, problemele NP-complete fiind o surpriză. Dar nu și pentru români. Ei au reușit să cucerească o medalie de aur, două de argint și una bronz.

Din secretele olimpiadelor

S-a constatat pe parcursul anilor că cele mai reușite organizări au fost realizate în acele țări unde există experiență în ceea ce privește organizarea concursurilor școlare de informatică. Nu este suficient ca interesul guvernamental să fie canalizat spre atingerea "culmilor civilizației" și *high-technology* la toate colțurile. Este nevoie de un interes special pentru cultivarea informaticii în rândurile tinerilor, pentru depistarea și "valorificarea" talentelor.

Anul acesta, în Portugalia, delegația României a fost invitată la ședința Comitetului Olimpic și a fost întrebată oficial dacă este de acord și, mai ales, dacă poate să organizeze o ediție IOI în viitor... Dincolo de bucuria și mândria firească, dintr-o dată am conștientizat că am intrat... "în rând cu lumea"! Nu se mai teme nimeni să vină la noi în țară, nu ne mai consideră nimeni lumea a treia sau, cel puțin, o țară săracă... Pentru că, trebuie spus un lucru esențial: o asemenea competiție se organizează și se realizează cu mulți bani (în funcție de prețurile din țara organizatoare, suma poate varia între 200.000 USD și 1 milion USD - apropos, cât e asta în lei?).

Pe de altă parte, peste tot unde s-a organizat până acum IOI, contribuția financiară a sponsorilor a fost consistentă. Sperăm că nu va fi altfel nici în România.

Viitorul apropiat

Constat că vă vorbesc despre un viitor îndepărtat și uit pentru moment de cel apropiat. În 2000 România organizează Olimpiada de Informatică a Europei Centrale. Pe lângă preocupările majore ale învățământului românesc, cu tot ce înseamnă ele în plină reformă, va trebui să găsim resurse umane și financiare pentru a demonstra pentru a nu știu câta oară că "Da! Suntem aici! Și suntem capabili să facem ceva pentru viitor!" Știu că nu doar olimpiadele sunt menite să demonstreze că și în România sunt tineri talentați, dar cred că nu avem voie să generalizăm neajunsurile. Lăsați-mă să vă arăt că există și *lumină* și că ambiția nu este neapărat un păcat.

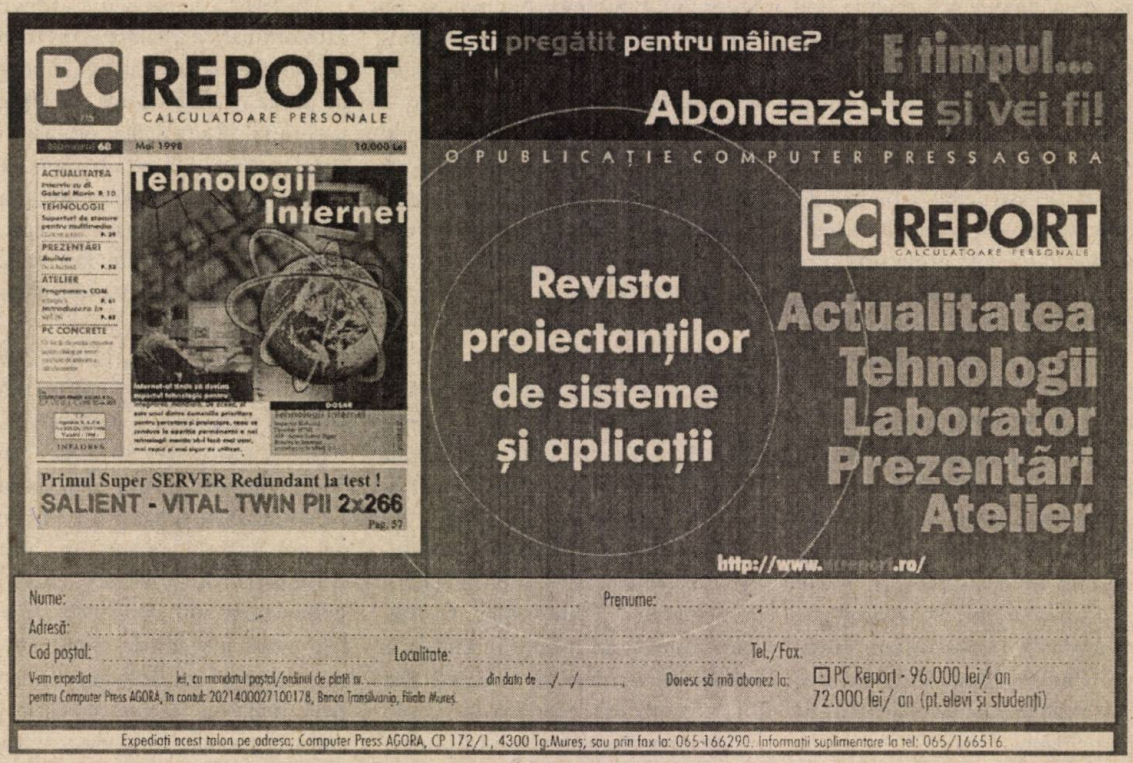
Ginfo în engleză?

Vă miră acest subtitlu? Poate vreți să mă întrebați: *Ce legătură are revista tinerilor informaticieni cu toate olimpiadele înșirate?* Vă răspund amintindu-vă generațiile de olimpici care, pe lângă multele surse de cunoștințe și de inspirație, au primit câte ceva și de la *Gazeta de Informatică* și, mai nou, de la *Ginfo*.

S-a întâmplat tot în Portugalia: m-au înconjurat câțiva profesori, conducători de delegații, și m-au întrebat: *Ce mai face Gazeta de Informatică?* Aflând cu bucurie că ea "trăiește" încă și există multe șanse să devină tot mai viguroasă (remarcați optimismul meu și încrederea că veți colabora tot mai consistent cu *GInfo*) au afirmat că ar fi extrem de bucuroși să poată oferi tinerilor din țările lor ceva similar...

Numai că în Grecia sau în Cipru sau în Iran nu există persoane care să aibă asemenea preocupări. Deci: vi se pare un proiect prea ambițios să editām... măcar patru numere pe an în limba engleză ?

Clara Ionescu



SOLUȚIILE problemelor propuse

Între 26 iulie - 9 august, la Constanța s-a desfășurat a doua tabără de antrenament a lotului olimpic național de informatică. La sfârșitul primei săptămâni membrii lotului lărgit au primit șase probleme la concursul organizat în scopul selecționării participanților la Olimpiada Internațională de Informatică. Vă prezentăm în continuare soluțiile problemelor.

P069801: Măsurătoare

Problemă propusă de prof. Eugen Ionescu, Liceul de Informatică, Cluj-Napoca

Rezolvarea propusă se bazează pe următoarele observații:

1. În cazul în care cantitatea care trebuie obținută nu este multiplu al celui mai mare divizor comun al capacităților celor două canistre, nu există soluție. Rezultă că n trebuie să dividă cmmdc(a, b);
2. Vom obține o soluție în cazul în care turnăm apa doar din prima canistră în cea de a doua;
3. Vom obține o soluție în cazul în care turnăm apa numai din a doua canistră în prima.

Soluția va fi aleasă dintre cele două soluții obținute pe baza observațiilor 2. și 3. Vom alege evident, soluția care implică cel mai mic număr de transferuri.

Listing CANISTRE.PAS

```
Program Canistre;
Var nr1,nr2,nr3,a,b,c,n,k:Longint;
    cont,lb:Longint;
    f:Text;
    gasit:Boolean;

  Function Cmmdc(a,b:Longint):Longint;
  begin
    if a*b=0
    then if a<>0 then Cmmdc:=a
                 else Cmmdc:=b
    else if a>b
         then Cmmdc:=Cmmdc(b,a-(a div b)*b)
         else Cmmdc:=Cmmdc(a,b-(b div a)*a)
  end;
Begin
  Assign(f,'CANA.IN'); Reset(f);
  Readln(f,n); Readln(f,a,b);
  Close(f);
  nr1:=0; nr2:=0;
  Assign(f,'CANA.OUT');
  if n=0
  then
  begin Rewrite(f); Close(f); Exit end;
  if n=b
  then
  begin
    Rewrite(f);
    Writeln(f,'UB');
    Close(f);
    Exit
  end;
  if (Cmmdc(a,b)<>1) and (n mod Cmmdc(a,b)<>0)
  then
  begin
    Rewrite(f);
    Writeln(f,'NU');
    Close(f);
    Exit
  end;
  c:=0;
  while c<>n do
  begin
    nr1:=nr1+2;
    if c+a>b then nr1:=nr1+2;
    c:=(c+a) mod b;
    if c=0 then c:=b
  end;
```

```
c:=0;
gasit:=false;
while (nr2<nr1) and not gasit do
begin
  nr2:=nr2+1;
  nr2:=nr2+2*((b+c) div a);
  nr2:=nr2+1;
  c:=(c+b) mod a;
  lb:=b+c;
  nr3:=0;
  while lb>n do
  begin
    lb:=lb-a;
    nr3:=nr3+1
  end;
  if lb=n
  then
  begin
    gasit:=true;
    nr2:=nr2+1+2*(nr3-1)+1
  end
end;
if nr2<nr1
then
begin
  Rewrite(f);
  c:=0; cont:=nr2;
  gasit:=false;
  while not gasit do
  begin
    Writeln(f,'UB');
    for k:=1 to (b+c) div a do
    begin
      Writeln(f,'BA'); Writeln(f,'GA')
    end;
    Writeln(f,'BA');
    c:=(c+b) mod a;
    lb:=b+c; nr3:=0;
    while lb>n do
    begin
      lb:=lb-a; nr3:=nr3+1
    end;
    if lb=n
    then
    begin
      gasit:=true;
      Writeln(f,'UB');
      for k:=1 to nr3-1 do
      begin
        Writeln(f,'BA'); Writeln(f,'GA')
      end;
      Writeln(f,'BA')
    end
  end;
  Close(f)
end
```

```
  else
  begin
    cont:=nr1;
    Rewrite(f);
    c:=0;
    while c<>n do
    begin
      Writeln(f,'UA');
      Writeln(f,'AB');
      if c+a>b
      then
      begin
        Writeln(f,'GB');
        Writeln(f,'AB')
      end;
      c:=(c+a) mod b;
      if c=0 then c:=b
    end;
    Close(f)
  end
End.
```

P069802: Nuclee

Problemă propusă de prof. Ioan Maxim, Liceul de Informatică, Suceava

Pentru început vom număra de câte ori apare fiecare număr cuprins între 1 și 65000 în șirul dat. Se observă că am avea nevoie de un tablou care să conțină 65000 de numere, fiecare număr necesitând o reprezentare pe 4 bytes. Cum limbajele Pascal și C nu permit declararea unei structuri de date de acest fel suntem nevoiți să alegem o altă reprezentare pentru această structură de date. Vom reprezenta șirul ca o matrice `A` de dimensiuni 256*256, fiecare linie a matricei fiind alocată dinamic. Celui de-al `i`-lea termen din șir îi va corespunde elementul `[i div 256, i mod 256]` din matrice.

Vom alege acum cei `k` termeni ai subșirului cerut. Pentru aceasta vom număra câte elemente care trebuie să aibă o anumită valoare trebuie să conțină acest subșir. Vom începe cu valoarea `m` și apoi, cât timp nu avem destule elemente vom căuta la stânga și la dreapta lui `m`, respectiv valorile: `m-1`, `m+1`, `m-2`, `m+2` etc. Trebuie să fim atenți ca numerele `m-i` sau `m+i` să fie în intervalul `[1..65000]`.

Pentru a afișa cei `k` termeni ai subșirului trebuie să mai parcurgem o dată șirul dat și, de fiecare dată când întâlnim un număr pentru care nu avem în subșir suficiente elemente cu acea valoare, îl adăugăm în subșir.

Listing NUCLEE.PAS

```
Program nuclee;
Const inputname='NUC.IN';
      outputname='NUC.OUT';
Type sir=array[0..255] of Longint;
Var f,g:Text;
    a:array[0..255] of ^sir;
    s,k,t:Longint;
```

```
Procedure Citire;
Var x:Word;
begin
  Assign(f,inputname); Reset(f);
  Readln(f,s,k);
  while not Seekeof(f) do
  begin
     Read(f,x);
     Inc(a[x div 256]^[x mod 256])
  end;
  Close(f)
end;

Procedure Initializari;
Var i:Byte;
begin
  for i:=0 to 255 do
  begin
    New(a[i]);
    Fillchar(a[i]^,Sizeof(a[i]^),0)
  end
end;

Procedure Calcul;
Var totaL:Word;
    i,decr,x,y:Longint;
begin
  total:=s;
  decr:=0;
  if a[k div 256]^[k mod 256]>total
  then a[k div 256]^[k mod 256]:=total;
  Dec(total,a[k div 256]^[k mod 256]);
  while total>0 do
  begin
    Inc(decr);
    x:=decr+k;
    y:=-decr+k;
    if x<65000
    then
    begin
      if a[x div 256]^[x mod 256]>total
      then a[x div 256]^[x mod 256]:=total;
      Dec(total,a[x div 256]^[x mod 256])
    end;
    if y>0
    then
    begin
      if a[y div 256]^[y mod 256]>total
      then a[y div 256]^[y mod 256]:=total;
      Dec(total,a[y div 256]^[y mod 256])
    end
  end;
  for i:=1 to 65000 do
    if Abs(i-k)>decr
    then a[i div 256]^[i mod 256]:=0
end;
```

```
Procedure Afisare;
Var x,c:Word;
begin
   Assign(f,inputname);
   Reset(f);
   Assign(g,outputname);
   Rewrite(g);
   Readln(f);
   c:=0;
   while not Seekeof(f) do
   begin
     Read(f,x);
     if a[x div 256]^[x mod 256]>0
     then
     begin
       Write(g,x,' ');
       Inc(c);
       if c mod 50 = 0 then Writeln(g);
       Dec(a[x div 256]^[x mod 256])
     end
   end;
   Close(g)
end;

Begin
  Initializari;
  Citire;
  Calcul;
  Afisare
End.
```

P069803: Electrotehnică

Problemă propusă de Ovidiu Gheorghioiu, student, Universitatea Politehnică, București

Din punct de vedere algoritmic problema este destul de simplă, singura dificultate fiind legată de determinarea unui sistem de bucle independente pentru *ecuațiile Kirchhoff 2*.

Metoda cea mai simplă de obținere a acestor L-N+1 bucle independente se bazează pe construirea unui arbore parțial al grafului dat. Acest arbore conține N-1 laturi și deci în afara lui rămân L-N+1 laturi. Fiecărei laturi exterioare îi putem atașa o buclă formată din latura respectivă și drumul între vârfurile ei prin interiorul arborelui. Buclele astfel obținute sunt într-adevăr în număr de L-N+1 și sunt independente deoarece fiecare conține o latură care nu se regăsește în nici una din celelalte bucle (vezi problema cu polițiștii dată la ONI Timișoara '97).

Sistemul este rezolvat prin *metoda Gauss*, astfel: matricea a memorează ecuațiile sistemului, fiecare linie conținând pe poziția 0 termenul liber (considerat în partea stângă a ecuației) iar pe celelalte poziții coeficienții intensităților corespunzătoare (exemplu: 2*I1+3*I2+5=0 se memorează sub forma: [5,2,3,0,0,....]). Rezolvarea constă în efectuarea pentru fiecare linie a următoarelor operații:

1) *găsirea unui element diferit de 0, exclusiv termenul liber;* există totdeauna un astfel de element deoarece în caz contrar, s-ar ajunge la o ecuație de forma `0+t1=0`. Dacă `t1` (termenul liber) este 0, înseamnă că am "pierdut" o ecuație, deci sistemul nu poate fi determinat.
2) *reducerea coloanei pe care se găsește acest element în celelalte ecuații (linii ale matricei).* Sistemul rezultat în urma acestor operații este foarte simplu, fiind format din ecuații de forma `I[x]*a[i,x]+t1=0` cu soluțiile evidente `I[x]=-t1/a[i,x]` cu `A[i,x]<>0`. Pentru ușurință vom reține `c[x]=i`. Sistemul se construiește destul de simplu pentru primele `N-1` ecuații (Kirchhoff I). Pentru ecuațiile Kirchhoff II se procedează ca mai sus.

Listing Electro.CPP

```
#include <fstream.h>
#include <stdlib.h>
#define Nmax 101
#define Lmax 101

double R[Lmax],E[Lmax],I[Lmax];
int va[Lmax],vb[Lmax],le[Lmax];
int m[Nmax],c[Nmax],ms[Nmax];
double *a[Lmax],b;
int N,L,i,j,k,ns,t,ne;
  void Citeste()
  {
    ifstream is("ELTH.IN", ios::in);
    is>>N>>L;
    for(i=1;i<=L;i++)
      is>>va[i]>>vb[i]>>R[i]>>E[i]>>le[i];
    is.Close();
  }

  int find(int k,int p)
// Gaseste un drum in arbore intre k si t
// (t global). Daca nu exista drum intoarce 0
// p = nodul precedent, pentru a evita
// intoarcerea pe aceeasi muchie
  {
    int l,i;
    if (k==t) return 1;
    for(i=1;i<N;i++)
      if (va[l=ms[i]] == k && vb[l] != p)
      if (find(vb[l],k))              //Gasit
        {
          a[ne][l]+=R[l];
                  //Adaug termenul R[l]*I[l]
          a[ne][le[l]]-=E[l];
    //Adaug termenul -E[l] la termenul liber,
          //respectiv termenul -E[l]*I[le[l]]
          return 1;
        } else;
      else
        if (vb[l]==k && va[l]!=p)
        if (find(va[l],k))
        {
         a[ne][l]-=R[l]; a[ne][le[l]]+=E[l];
         return 1;
        }
      return 0;
  };

  void Sistemul()
// Construieste sistemul de ecuatii Kirchhoff
  {
    for (i=1; i<=L; i++)
    {
      if (!(a[i] = New double[L+1]))
      {
        cerr << "Memorie insuficienta!";
        exit(1);
      };
      memset(a[i], 0, (L+1) * Sizeof(double));
    }

    //K1 : Primele N-1 ecuatii: latura i iese
    //din va[i] si intra in vb[i]
    for (i=1; i<=L; i++)
    { a[va[i]][i]=1; a[vb[i]][i]=-1;
    }
    memset(a[N],0,(L+1)*Sizeof(double));
                        //Ultima nu mai conteaza
    for(i=1;i<=N;i++) c[i]=i;
//Arborele partial: Kruskal neoptimizat (100 max.)
    for (i=1;i<=L;i++)
      if (c[va[i]]!=c[vb[i]])
      {
        ms[++ns]=i;
         // ns:numarul de muchii selectate;
         // ms: muchiile sel.
        k=c[vb[i]];
        for (j=1;j<=N;j++)
          if (c[j]==k) c[j]=c[va[i]];
        if (ns==N-1) break;
      };
      // K2
      ns = 1; ne = N;  //ne = Ecuatia curenta
      for (i=1;i<=L;i++)
        if (i==ms[ns]) ns++;
          // Sarim peste muchiile din arbore
        else
        {
          //Gasesc un ciclu
          t=vb[i]; // t = "tinta"
          find(va[i],0);
          a[ne][i]-=R[i];
        //Adaug latura i, insa in sens invers
          a[ne][le[i]]+=E[i]; ne++;
        }
  }
```

```
 void Rezolva()
 {
   memset(c, 0, Sizeof(c));    //"Reciclez" c
                                        // Gauss
   for(i=1;i<=L;i++)
   {
     for(j=1;j<=L;j++)
       if (a[i][j]) {t=j; j=L;}
                        //t exista intotdeauna
     c[t]=i;
    //I[t] este det. de ecuatia cu numarul i
     for (j=1; j<=L; j++)
                    //Reduc celelalte ecuatii
       if (j!=i)
       {
         b=-a[j][t]/a[i][t];
         for(k=0;k<=L;k++)
           a[j][k]+=b*a[i][k];
//se observa ca elementul a[j][t] a devenit 0
       }
   }
   for(i=1;i<=L;i++)
     I[i]=-a[c[i]][0]/a[c[i]][i];
 }
 void Scrie()
 {
   ofstream os("elth.out");
   os.precision(2);
   for(i=1;i<=L;i++) os<<I[i]<<"\n";
   os.Close();
 }

void main()
{
  Citeste();
  Sistemul();
  Rezolva();
  Scrie();
}
```

P069804: Perechi stabile

Problemă propusă de prof. Emil Onea, Liceul "Unirea", Focșani

Schițăm pașii algoritmului propus:

- variabila `b`, inițializată cu valoarea 1, denumește bărbatul pentru care se va stabili perechea în momentul curent;
- se parcurg descrescător preferințele bărbatului `b`; să presupunem că s-a ajuns la femeia `f`:
  - dacă aceasta este necuplată, se cuplează `b` cu `f` și `b` se incrementează cu 1 (trecem la următorul bărbat);
  - dacă aceasta este cuplată:
    - dacă ea preferă bărbatul `b` în locul actualului ei bărbat, se cuplează `b` cu `f` și `b` se setează pe valoarea fostului bărbat al femeii `f`;
    - altfel, se continuă baleierea preferințelor bărbatului `b`.

Trebuie demonstrat că algoritmul *este finit* și că *produce un rezultat corect*.

Să observăm că o femeie, odată cuplată, nu va mai fi niciodată liberă, eventual își schimbă perechea. Astfel, constatăm că la fiecare pas, fie o nouă femeie este cuplată, fie o femeie deja cuplată își schimbă partenerul cu unul aflat mai în față în lista preferințelor ei. Deci la fiecare pas, configurația generală se "îmbunătățește". Este evident că acest proces nu se poate continua la infinit, iar în situația cea mai avantajoasă se ajunge la o configurație în care fiecare femeie este cuplată cu primul bărbat în ordinea preferințelor ei. Rezultă deci că algoritmul este finit; mai mult, deducem limita superioară a complexității sale: deoarece o femeie își poate schimba bărbatul de cel mult `N-1` ori, iar numărul femeilor este `N`, tragem concluzia că algoritmul efectuează cel mult `N*(N-1)` pași; complexitatea sa este deci $O(N^2)$.

Să demonstrăm acum că algoritmul produce un rezultat corect. Considerăm "tabela" cuplărilor, produsă după execuția algoritmului și perechea arbitrară `(b,f)`. Să presupunem că ar exista o femeie `f'`, pe care `b` o preferă față de `f`, și care la rândul ei îl preferă pe `b` în fața bărbatului ei; fie acesta `b'`. Vom demonstra că acest lucru este însă imposibil. Într-adevăr, atunci când am făcut alegerea femeii care devine perechea lui `b`, `f'` a fost tratată înaintea lui `f`, conform presupunerii făcute. Apar 2 cazuri:

- dacă `f'` era necuplată în acel moment, atunci `b` s-ar fi cuplat cu `f'`; `b` fiind, după preferințele lui `f'`, în fața lui `b'`, acesta n-ar fi ajuns niciodată să se cupleze cu `f'` (am observat că o femeie își "îmbunătățește" alegerea sau o lasă constantă) - contradicție, deoarece în tabela finală `b'` este cuplat cu `f'`;
- dacă `f'` era cuplată în acel moment, atunci era cuplată cu `b'` sau cu un alt bărbat, `b"`, aflat, după preferințele ei, după `b'`; oricum, în "clasamentul" lui `f'`, `b` era în fața bărbatului din acel moment al lui `f'`, deci, conform algoritmului, `f'` s-ar fi despărțit de bărbatul ei, cuplându-se cu `b`. Din nou, `b` fiind, după `f'`, în fața lui `b'`, `b'` nu s-ar mai fi putut cupla cu `f'` - contradicție.

Cum alegerea perechii `(b, f)` a fost făcută arbitrar, este clar că tabela finală nu poate cuprinde decât perechi stabile.

Observație

Soluția în general nu este unică.

Exemplu:

Fie 6 bărbați și 6 femei pentru care avem următoarele preferințe:

Preferințele bărbaților :	Preferințele femeilor :
1: 4 5 2 3 1 6	1: 4 2 3 6 5 1
2: 3 5 2 4 6 1	2: 3 2 4 5 6 1
3: 3 4 6 5 1 2	3: 1 5 4 6 2 3
4: 5 4 2 3 1 6	4: 1 3 2 4 6 5
5: 6 3 1 4 5 2	5: 5 3 4 6 1 2
6: 1 2 3 6 5 4	6: 4 5 1 6 2 3

O soluție ar fi:
(1, 4);
(2, 3);
(3, 5);
(4, 2);
(5, 6);
(6, 1);

O altă soluție:
(1, 4);
(2, 2);
(3, 5);
(4, 3);
(5, 6);
(6, 1);

Listing PERECHI.PAS

```
Program Perechi_stabile;
const fi='PAIRS.IN';
      fo='PAIRS.OUT';
      nmax=250;

type linie=array[1..nmax] of Integer;
     plinie=^linie;
     tablou=array[1..nmax] of plinie;

var mpref,fclas:tablou;
    fem,barb:linie;
    b,n,i,j,k,w,cuplat:Integer;
    f:Text; gata:Boolean;

procedure citire;
begin
  Assign(f,fi);
  Reset(f);
  Readln(f,n);
  for i:=1 to n do
  begin
    New(mpref[i]);
    for j:=1 to n do
      Read(f,mpref[i]^[j]);
    Readln(f)
  end;
  Readln(f);
  for i:=1 to n do
  begin
    New(fclas[i]);
    for j:=1 to n do
    begin
      Read(f,k);
      fclas[i]^[k]:=j
    end;
    Readln(f)
  end;
  Close(f)
end;

procedure initializari;
begin
  for i:=1 to n do
  begin
    fem[i]:=0;
    barb[i]:=0
  end
end;

procedure perechi;
begin
  b:=1;
  while fem[b]<>0 do b:=b+1;
  i:=1;
  gata:=false;
  repeat
    w:=mpref[b]^[i];
    if barb[w]<>0
    then
    begin
      if fclas[w]^[b]<fclas[w]^[barb[w]]
      then gata:=true
    end
    else gata:=true;
    if not gata then i:=i+1
  until gata;
  if barb[w]<>0 then fem[barb[w]]:=0
                else cuplat:=cuplat+1;
  fem[b]:=w; barb[w]:=b
end;

procedure scrie;
begin
  Assign(f,fo); Rewrite(f);
  for i:=1 to n do Writeln(f,i,' ',fem[i]);
  Close(f)
end;

Begin
  citire;
  initializari;
  repeat perechi until cuplat=n;
  scrie
End.
```

P069805: Wilhelm Tell

Problemă propusă de Cătălin Frâncu, student Universitatea Politehnică, București

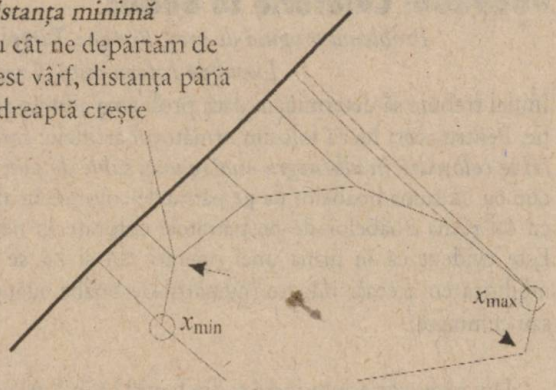
Se observă că pot exista două situații:

1. dreapta atinge poligonul, caz în care abaterea este 0;
2. dreapta nu atinge poligonul, caz în care distanța de la poligon la dreaptă este practic minimul distanței de la unul din vârfurile poligonului la dreaptă.

De aceea, o soluție banală ar fi un algoritm de complexitate $O(M*N)$ care face testul pentru fiecare dreaptă în $O(N)$: dacă există vârfuri ale poligonului situate de ambele părți ale dreptei (sau chiar pe dreaptă), distanța este 0, altfel se află minimul distanței de la vârfurile poligonului la dreaptă.

Un algoritm de complexitate $O(M*\log N)$ face testul pentru fiecare dreaptă în $O(\log N)$ și se bazează pe următoarea observație: dacă divizăm poligonul în două lanțuri poligonale între vârfurile cu coordonata x minimă și maximă, atunci pentru fiecare lanț funcția "distanța de la vârf la dreaptă" este întâi descrescătoare, apoi crescătoare, ca în imaginea următoare:

Distanța minimă
Cu cât ne depărtăm de acest vârf, distanța până la dreaptă crește



Deducem că vârful situat cel mai aproape se determină prin bisecția intervalului. Tot în timpul bisecției verificăm dacă nu se sare dincolo de dreaptă, caz în care dreapta este secantă.

Programul caută extremele pe ambele lanțuri poligonale. În figură vârfurile descoperite sunt marcate cu săgeată. Dacă aceste puncte se află de o parte și de alta a dreptei, dreapta este secantă. Altfel, unul dintre ele trebuie să fie cel mai apropiat de dreaptă din tot poligonul.

Merită remarcat că dacă un vârf este mai aproape de dreaptă decât ambii săi vecini, atunci el este punctul cel mai apropiat de dreaptă din întregul poligon.

Listing WTELL.PAS

```
Program Tell;
{$B-,F+,I-,N+,R-,S-}
Const NMax=3000;
Type PointType=record X,Y:Extended end;
     PointVector=array[-1..NMax]of PointType;
     BoolFunc=Function(i,j:Integer):Boolean;
Var P:PointVector;
    N,M,NSecants,IndMinX,IndMaxX:Integer;
          { Indicii punctelor extreme pe X }
    A,B,C,SumDist:Extended;
    Better:BoolFunc;  { Criteriul folosit   }
             { la maximizare (Greater/Less) }

  Procedure ReadPoints;
      { Citeste punctele si calculeaza      }
      { indicii minimului & maximului pe X }
  Var i:Integer;
  begin
    Readln(N); IndMinX:=0; IndMaxX:=0;
    for i:=0 to N-1 do
      with P[i] do
      begin
        Readln(X,Y);
        if X<P[IndMinX].X then IndMinX:=i;
        if X>P[IndMaxX].X then IndMaxX:=i
      end;
    NSecants:=0; SumDist:=0
  end;

  Function Dist(i:Integer):Extended;
              { Distanta de la un punct     }
              { la o dreapta (normalizata) }
  begin
    Dist:=A*P[i].X+B*P[i].Y+C
  end;

  Function Greater(i,j:Integer):Boolean;

     { Arata daca P[i] este "mai mare"      }
     { ca P[j] relativ la directia dreptei }
  begin
    Greater:=(Dist(i)>=Dist(j))
  end;

  Function Less(i,j:Integer):Boolean;
     { Arata daca P[i] este "mai mic"       }
     { ca P[j] relativ la directia dreptei }
  begin
    Less:=(Dist(i)<=Dist(j))
  end;

  Function FindBest(Lo,Hi:Integer;
                    Better:BoolFunc):Integer;
{ Intoarce indicele punctului optim pe      }
{ directia vectorului de componente A si B.}
{ "Optim" inseamna fie "maxim", fie "minim"}
{ dupa cum Better este Greater sau Less     }
{ (criteriul de comparatie)                 }
  Var Mid:Integer;
      Finished:Boolean;
  begin
    Finished:=False;
   while((Hi+N-Lo) mod N>1) and not Finished do
    begin
      if Hi>Lo
      then Mid:=(Lo+Hi) div 2
      else Mid:=((Lo+Hi+N) div 2) mod N;
                                { Sare peste 0 }
      if Better(Mid,(Mid+1) mod N)
      then Hi:=Mid
      else
        if Better(Mid,(Mid+N-1) mod N)
        then Lo:=Mid
        else Finished:=True
                     { Suntem intr-o groapa }
    end;
    if Better(Hi,Lo) then FindBest:=Hi
                     else FindBest:=Lo
  end;

  Procedure CountDistance(Min,Max:Integer);
{ Vede daca minimul si maximul gasite dupa  }
{ directia ceruta sunt:                     }
{   a) De parti diferite ale dreptei ->     }
{      dreapta e secanta, distanta e zero   }
```

```
{   b) De aceeasi parte, distanta este a    }
{      celui mai apropiat dintre ele        }
  Var DMin, DMax:Extended;
  begin
    DMin:=Dist(Min);
    DMax:=Dist(Max);
    if DMin*DMax<=0
    then Inc(NSecants)
    else
      if Abs(DMin)<Abs(DMax)
      then SumDist:=SumDist+Abs(DMin)
      else SumDist:=SumDist+Abs(Dmax)
  end;

  Procedure CheckLines;
  Var i:Integer;
      Max,Min,Candidate:Integer;
      D:Extended;
  begin
    Readln(M);
    for i:=1 to M do
    begin
      Readln(A,B,C);
              { Normalizam ecuatia dreptei }
      D:=Sqrt(A*A+B*B);
      A:=A/D;
      B:=B/D;
      C:=C/D;
      { Cautam punctul MAXIM dupa directia }
      { data, pe ambele lanturi            }
      Max:=FindBest(IndMinX,IndMaxX,Greater);
 Candidate:=FindBest(IndMaxX,IndMinX,Greater);
      if Greater(Candidate,Max)
      then Max:=Candidate;
      { Cautam punctul MAXIM dupa directia }
      { data, pe ambele lanturi            }
      Min:=FindBest(IndMinX,IndMaxX,Less);
      Candidate:=FindBest(IndMaxX,IndMinX,Less);
      if Less(Candidate,Min)
      then Min:=Candidate;
      CountDistance(Min,Max)
    end
  end;

Begin
  Assign(Input,'TELL.IN');
  Reset(Input);
  Assign(Output,'TELL.OUT');
  Rewrite(Output);
  ReadPoints;
  CheckLines;
  Writeln(NSecants);
  Writeln(SumDist:0:2);
  Close(Input);
  Close(Output)
End.
```

P069806: Călătorie cu bucluc

Problemă propusă de prof. Roxana Tâmplaru, Liceul de Informatică, Craiova

Inițial trebuie să determinăm dacă problema admite soluție. Pentru acest lucru folosim următorul artificiu: *caroiajul se colorează în alb/negru analog unei table de șah*; notăm cu SA suma boabelor de pe pătratele colorate în alb și cu SN suma boabelor de pe pătratele colorate în negru. Este evident că în urma unei operații SA și SN se vor modifica cu aceeași valoare (numărul de boabe adăugate sau eliminate).

Din propoziția anterioară și din faptul că în final caroiajul va fi gol, rezultă că existența soluției este garantată de egalitatea SA=SN. De asemenea dacă există soluție, în urma oricărei succesiuni de operații SA va fi egal cu SN. Deci vom calcula sumele SA și SN și dacă ele diferă, rezultă că nu există soluție (funcția Posibil) și se va scrie 0 în fișierul de ieșire. Dacă rezultă că există soluție trebuie găsită succesiunea de operații care să elimine toate boabele de cafea. Pentru a o determina vom parcurge căsuțele caroiajului în următoarea ordine:

(1,1)(1,2)(1,3)...(1,n)(2,n)(2,n-1)...(2,1)(3,1)(3,2)...

Observații:

- fiecare căsuță este vizitată o singură dată;
- oricare două căsuțe vizitate succesiv sunt vecine (adică deplasarea se face pe linie sau coloană);
- ultima căsuță vizitată este (m, 1) dacă n este par sau (m, n) în caz contrar (n impar).

Se parcurg primele m*n-2 căsuțe (în ordinea amintită) și se elimină, din fiecare, toate boabele, astfel:

- dacă numărul de boabe este 0, atunci se trece la următoarea căsuță;
- dacă numărul de boabe din căsuța curentă este mai mic sau egal decât cel din căsuța următoare (în ordinea parcurgerii) atunci se va elimina din căsuța curentă și din următoarea un număr de boabe egal cu cel din căsuța curentă (astfel se va goli căsuța curentă);
- dacă numărul de boabe din căsuța curentă este mai mare decât cel din căsuța următoare, numărul de boabe din căsuța următoare se va mări astfel încât să ajungă egal cu cel din căsuța curentă (se adaugă boabe la căsuța următoare și succesoarei acesteia). Se va elimina din căsuța curentă și din cea următoare un număr de boabe egal cu cel din căsuța curentă; astfel se vor goli cele două căsuțe.

După aceste operații vor mai exista cel mult două căsuțe nevide (ultimele două). Cum cele două căsuțe sunt vecine, ele sunt colorate diferit. Din faptul că în orice moment SA=SN rezultă că în cele două căsuțe vor exista același număr de boabe, care se vor elimina.

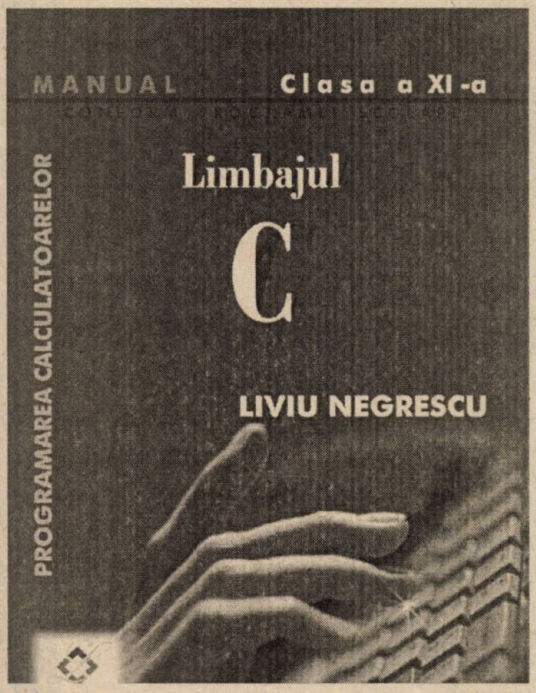
Listing Calatorie.PAS

```
Program CalatorieCuBucluc;
type Sir=array[1..150] of Longint;
Var m,n:Byte;
    suma:Longint;
    fo:Text;
    a:array[1..150] of ^Sir;

  Procedure Citire;
  Var f:Text;
      i,j:Byte;
  begin
    Assign(f,'IN.TXT');
    Reset(f);
    Readln(f,m,n);
    for i:=1 to m do
    begin
      New(a[i]);
      Fillchar(a[i]^,Sizeof(a[i]^),0)
    end;
    for i:=1 to m do
      for j:=1 to n do
        Read(f,a[i]^[j]);
    Close(f)
  end;

  Function Posibil:Boolean;
        { decide daca problema are solutie }
  var s1,s2:Longint;
      i,j:Byte;
  begin
    s1:=0; s2:=0;
    for i:=1 to m do
      for j:=1 to n do
        if Odd(i+j)
        then Inc(s1,a[i]^[j])
        else Inc(s2,a[i]^[j]);
    Posibil:=s1=s2
  end;

  Procedure Urmator(x,y:Byte; var xx,yy:Byte);
  begin
    xx:=x; yy:=y;
    if Odd(x)
    then
      if yy=n
      then
      begin
        xx:=x+1;
        yy:=n
      end
      else yy:=y+1
    else
      if yy=1
      then
      begin
        xx:=x+1;
        yy:=1
      end
      else yy:=y-1
  end;

  Procedure Mutare(x1,y1:Byte; k:Longint);
  var x2,y2:Byte;
  begin
    if k=0 then Exit;
    Urmator(x1,y1,x2,y2);
    Inc(a[x1]^[y1],k);
    Inc(a[x2]^[y2],k);
    Writeln(fo,x1,',',y1,' ',x2,',',y2,'->',k)
  end;

  Procedure Rezolvare;
  var x,y,xx,yy:Byte;
      k:Longint;
  begin
    x:=1;
    y:=1;
    while x<>m+1 do
    begin
      Urmator(x,y,xx,yy);
      if a[x]^[y]>a[xx]^[yy]
      then
      begin
        k:=a[x]^[y]-a[xx]^[yy];
        Mutare(xx,yy,k)
      end;
      Mutare(x,y,-a[x]^[y]);
      x:=xx;
      y:=yy
    end
  end;

Begin
  Citire;
  Assign(fo,'OUT.TXT');
  Rewrite(fo);
  if Posibil
  then Rezolvare
  else writeln(fo,0);
  Close(fo)
End.
```

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Această carte este concepută în conformitate cu Programa analitică a clasei a XI-a, profilul Informatică. Ea poate fi considerată o introducere elementară în limbajul C. Fiecare noțiune este explicată în mod clar și utilizată în programe simple. De asemenea, capitolele sunt însoțite de exerciții propuse care contribuie la fixarea și înțelegerea noțiunilor întâlnite în capitolele respective.

Cartea se adresează atât elevilor cât și celor interesați să se inițieze în limbajul C care în prezent este unul dintre cele mai utilizate limbaje de programare.

Liviu Negrescu este cercetător principal gr. I la societatea SOFTWARE ITC S.A., filiala Cluj, unde se ocupă cu dezvoltări de programe de sistem și aplicative.
În același timp, desfășoară o activitate didactică prestigioasă de peste 15 ani, la Facultatea de Calculatoare de la Universitatea Tehnică din Cluj.
În decursul anilor a fost titular a mai multe cursuri, ca de exemplu:

- Limbaje de programare și compilatoare;
- Proiectarea translatoarelor etc.

Liviu Negrescu are o serie de lucrări științifice publicate în diferite reviste de specialitate. În colaborare cu alți cercetători de la SOFTWARE ITC S.A. a implementat sisteme cu aplicații în diferite domenii:

- Construcția de compilatoare;
- Interfețe utilizator;
- Prelucrări de date din domenii economice și medicale.

Liviu Negrescu este autorul mai multor cărți apărute în decursul anilor, cu privire la diferite limbaje de programare (CPL1-subset al limbajului PL/1, CDL-limbaj de descriere al compilatoarelor precum și referitoare la limbajele C și C++. Lucrarea beneficiază de experiența vastă a autorului câștigată atât în activitatea de cercetare cât și în cea didactică.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

PROBLEME propuse pentru REZOLVARE

Între 26 iulie - 9 august, la Constanța s-a desfășurat a doua tabără de antrenament a lotului olimpic național de informatică. La sfârșitul celei de a doua săptămâni concurenții au primit șapte probleme. Adresându-ne tuturor celor care iubesc algoritmica, vă propunem să vă încercați puterile!

P079801: Coloane

Problemă propusă de conf. dr. Henri Luchian, Universitatea "Al. I. Cuza", Iași

Se dă o matrice având elemente din mulțimea {0,1}. Să se selecteze un număr minim de coloane din această matrice astfel încât suma elementelor pe fiecare linie, luând doar elementele din coloanele selectate, să fie cel puțin 1.

Date de intrare

Pe prima linie a fișierului de intrare `COLOANE.IN` se află două numere naturale `m` și `n` ($1 \le m, n \le 200$), despărțite printr-un singur spațiu, reprezentând numărul de linii și de coloane ale matricei.

Pe următoarele `m` linii se află elementele matricei; două elemente de pe aceeași linie sunt despărțite printr-un singur spațiu.

Date de ieșire

În fișierul de ieșire `COLOANE.OUT` se vor scrie datele de ieșire sub forma următoare:

Pe prima linie se va scrie numărul minim de coloane selectate.

Pe următoarea linie se vor scrie numerele de ordine ale coloanelor selectate, despărțite printr-un singur spațiu.

Dacă nu există soluție, atunci prima linie va conține un singur număr 0.

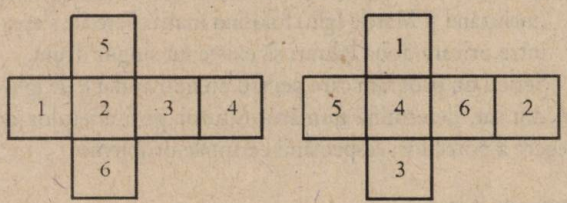
Exemplu:

COLOANE.IN	COLOANE.OUT
6 5	2
1 0 1 0 0	1 3
0 1 1 1 0	
1 1 1 0 1	
0 0 1 0 0	
1 0 0 1 1	
1 0 0 0 1	

P079802: Cuburi

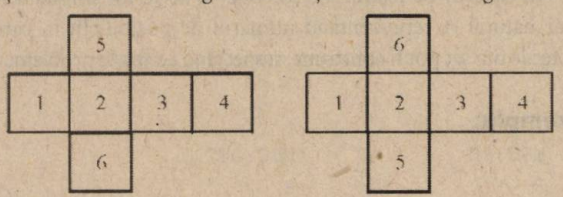
Problemă propusă de profesorii Maria și Adrian Nițâ, Liceul Emanoil Gojdu, Oradea

Vom considera un mod de desfășurare a unui cub, în care fiecare față a cubului este numerotată cu o cifră de la 1 la 6. Desfășurarea cubului se poate realiza în mai multe moduri. Avem, de exemplu următoarele două moduri de desfășurare diferite ale aceluiași cub:



O desfășurare se descrie printr-o secvență de șase cifre, citite de sus în jos și pe fiecare linie, de la stânga la dreapta. Deci pentru exemplele date vom avea succesiunea de cifre `512346` respectiv `154623`.

Cele două desfășurări de mai jos nu provin de la același cub deoarece imaginile desfășurărilor sunt "oglindite".



Scrieți un program care să distingă între două perechi de succesiuni de cifre date dacă sunt sau nu identice, în sensul că reprezintă desfășurarea aceluiași cub sau nu.

Date de intrare

Fișierul `CUB.IN` conține mai multe linii. Pe fiecare linie sunt scrise câte două perechi de succesiuni de cifre, formate din câte șase cifre distincte; succesiunile de șase cifre sunt despărțite printr-un singur spațiu.

Date de ieșire

Fișierul de ieșire CUB.OUT va conține atâtea linii câte perechi de numere au fost în fișierul de intrare. Pe fiecare linie se va scrie cuvântul 'DA', respectiv 'NU' în funcție de constatarea că cele două desfășurări corespund sau nu aceluiași cub.

Exemplu:

CUB.IN	CUB.OUT
512346 154623	DA
512346 612345	NU

P079803: Eschimoși

Problemă propusă de lect. Clara Ionescu, Universitatea "Babeș-Bolyai", Cluj

Eschimoșii au așezări în care își petrec viața, locuind în igluuri. În unele sate toate **igluurile sunt așezate pe aceeași linie** și astfel este suficient să existe cărare doar între oricare două igluuri vecine (spre stânga și/sau spre dreapta).

După ce Marele Eskimo a devenit regele tuturor eschimoșilor el a cerut poporului său să construiască în fiecare sat un Mare Iglu care să nu se afle pe aceeași linie de-a lungul căreia sunt așezate igluurile sătenilor. De la acesta trebuia să se croiască potecă la fiecare iglu din sat. Marele Eskimo și-a dat seama că este prea greu să-și refacă potecile după fiecare furtună de zăpadă. A ales un număr minim de poteci astfel încât:

- să fie posibil să se ajungă de la un iglu la oricare altul (incluzând și Marele Iglu) folosind numai potecile alese;
- între oricare două igluuri să existe un singur drum.

Scrieți un program care pentru un număr dat N de igluuri din sat, determină numărul tuturor posibilităților de alegere a potecilor, respectând cerințele problemei.

Date de intrare

În fișierul de intrare ESC.IN este scris un singur număr natural N ($2 \le N \le 35$), reprezentând numărul igluurilor din sat (construite de-a lungul unei linii).

Date de ieșire

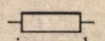
În fișierul de ieșire ESC.OUT se va scrie un singur număr natural M, reprezentând numărul de posibilități în care potecile din sat pot fi construite, respectând cerințele problemei.

Exemplu:

ESC.IN	ESC.OUT
3	8

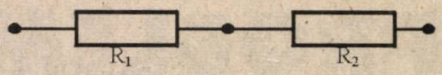
P079804: Rezistențe

Problemă propusă de Valentin Gheorghiță, student, Universitatea Politehnică, București

Rezistoarele sunt componente electrice cu două capete (noduri de rețea). Un rezistor este simbolizat de obicei în felul următor: 

Nu există noțiunea de capăt de intrare sau ieșire; totul depinde de modul în care circulă curentul electric. Deci un rezistor este simetric și bidirecțional.

Legarea în serie a două rezistoare este realizată astfel:



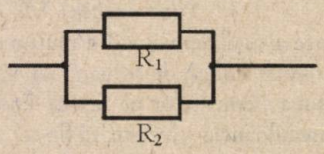
Rezultatul acestei legări în serie este echivalentul unui rezistor *Re* a cărui valoare este suma rezistențelor celor două componente (pentru exemplu $Re=R_1+R_2$).

Între cele două rezistoare se consideră un singur nod, comun. În desen, nodurile sunt reprezentate astfel: •

Rezultatul legării în paralel a două rezistoare este echivalentul unui rezistor al cărui valoare este calculată după formula (dacă cele două rezistoare componente au valorile R_1 respectiv R_2):

$$Re = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}}$$

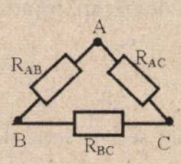
Legarea în paralel a două rezistoare este realizată astfel:

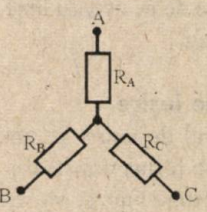


În circuitele pe care le veți studia veți avea nevoie și de transformări de tip triunghi-stea. Aceasta este următoarea:

$$R_A = \frac{R_{AB} \cdot R_{AC}}{R_{AB} + R_{AC} + R_{BC}} \qquad R_B = \frac{R_{AB} \cdot R_{BC}}{R_{AB} + R_{AC} + R_{BC}}$$

$$R_C = \frac{R_{BC} \cdot R_{AC}}{R_{AB} + R_{AC} + R_{BC}}$$





Problema constă în a găsi rezistențele echivalente pentru diverse rețele de rezistoare, știind că sunt necesare doar cele trei transformări prezentate (serie, paralel, triunghi-stea) și combinații ale acestora.

Scrieți un program care calculează rezistența echivalentă între două noduri ale unui circuit dat.

Date de intrare

Fișierul de intrare REZ.IN are următoarea structură:

- pe prima linie este scris numărul N, reprezentând numărul de noduri ale rețelei;
- pe a doua linie se află o pereche de numere, reprezentând nodurile între care trebuie calculată rezistența echivalentă;
- pe următoarele linii (mai puține decât 5001) sunt scrise triplete de forma i j k având semnificația: între nodu-

rile `i` și `j` se află un rezistor de valoare `k`; cele trei numere sunt despărțite prin câte un spațiu.

Observații:

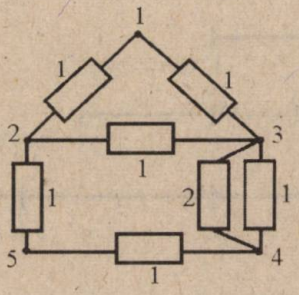
1. Toate numerele primite din fișierul de intrare sunt naturale, mai mici decât 5001.
2. Dacă între nodurile `i` și `j` sunt mai multe rezistențe atunci ele sunt legate în paralel.

Date de ieșire

Valoarea rezistenței echivalente se va scrie în `REZ.OUT`, cu 3 zecimale exacte.

Exemplu:

```
REZ.IN
5
1 5
1 2 1
1 3 1
2 3 1
3 4 1
3 4 2
4 5 1
2 5 1
REZ.OUT
1.133
```



P079805: Seiful

Problemă propusă de prof. Dana Lica, Liceul "I. L. Caragiale", Ploiești

Se consideră o comisie formată din `n` persoane (numerotate de la 1 la `n`) care trebuie să păstreze într-un seif subiectele de la examenul de admitere.

Să se determine numărul minim de lacăte `lmin` necesar închiderii seifului astfel încât să existe o distribuție a cheilor lor care să îndeplinească următoarele condiții:

- oricare două persoane dețin același număr de chei;
- fiecare persoană deține chei de la lacăte diferite;
- toate lacătele seifului se vor putea deschide numai în prezența oricărui grup format din cel puțin `n-1` persoane.

Observații:

1. Pentru un lacăt pot exista mai multe chei care să-l deschidă.
2. Nici o cheie nu poate deschide două lacăte diferite.
3. Lacătele sunt numerotate de la 1 la `lmin`.

Să se determine numărul minim de lacăte, numărul total de chei distribuite și o repartizare a cheilor care respectă condițiile problemei.

Date de intrare

În fișierul de intrare `SEIF.IN` se va găsi numărul `n` (`n≤500`) al membrilor comisiei.

Date de ieșire

În fișierul de ieșire `SEIF.OUT` se vor scrie datele de ieșire sub forma următoare:

- pe prima linie se vor scrie două numere naturale separate printr-un singur spațiu:

  `lmin chei`

  unde:

  `lmin` reprezintă numărul minim de lacăte;

  `chei` reprezintă numărul total de chei (de la toate cele `lmin` lacăte);

- pe fiecare din următoarele `lmin` linii se va scrie câte o secvență de numere naturale, separate printr-un singur spațiu. Pe linia `i+1` se va afla șirul numerelor de ordine a membrilor care primesc câte o cheie de la lacătul `i`.

În cazul în care există mai multe soluții, se cere una singură.

Obsevație:

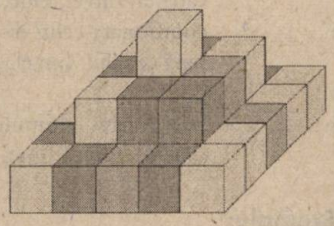
Dacă programul furnizează numărul corect pentru `lmin` și `chei` se va acorda aproximativ 25% din punctajul pentru testul respectiv.

Exemplu:

```
SEIF.IN          SEIF.OUT
3                3 6
                 3 2
                 1 2
                 3 1
```

P07986: Piramida lui Keops

Problemă propusă de Marius Vlad, student, Universitatea Politehnică, București



În piramida lui Keops trebuie săpat un canal de la bază și până în vârful piramidei.

Piramida este construită din blocuri de piatră cubice de duritați diferite. Baza sa (nivelul 0) este un pătrat și este alcătuită din `N*N` blocuri de piatră unde `N` este un număr natural impar (`0<N<30`). Fiecare nivel este perfect centrat peste nivelul inferior și are o lungime cu 2 blocuri de piatră mai mică decât lungimea blocului inferior. Ultimul nivel este format dintr-un singur bloc de piatră.

Duritatea fiecărui bloc de piatră este un număr natural din intervalul `0..200`.

Pentru săparea canalului trebuie respectate mai multe condiții:

- În primul rând munca depusă la săparea acestui canal trebuie să fie minimă. Înțelegem prin munca depusă suma durităților blocurilor prin care trece canalul.
- Canalul trebuie început de pe oricare latură exterioară a bazei.
- Canalul trebuie terminat în vârf (pe blocul de piatră de pe cel mai înalt nivel al piramidei).
- Canalul nu poate coborî (pe un nivel inferior).

- Între oricare două puncte de pe același nivel lungimea canalului trebuie să fie minimă. Înțelegem prin lungimea canalului între două puncte de pe același nivel numărul de blocuri de piatră prin care trece canalul (între cele două puncte specificate de pe același nivel).
- La trecerea dintr-un bloc de piatră în altul canalul trece prin fața comună celor două blocuri.

Datele de intrare

Fișierul text PIRAMIDA.IN cu următoarea structură:

N - lungimea bazei

$D_{0,1,1}$ $D_{0,1,2}$ ... $D_{0,1,n}$ - duritatea liniei 1 din bază

$D_{0,2,1}$ $D_{0,2,2}$ ... $D_{0,2,n}$ - duritatea liniei 2 din bază

...

$D_{0,n,1}$ $D_{0,n,2}$ ... $D_{0,n,n}$ - duritatea liniei n din bază

(*baza* (*nivelul* 0) *are* n *linii și* n *coloane*)

$D_{1,1,1}$ $D_{1,1,2}$ ... $D_{1,1,n.2}$ - duritatea liniei 1, nivelul 1

$D_{1,2,1}$ $D_{1,2,2}$ ... $D_{1,2,n-2}$ - duritatea liniei 2, nivelul 1

...

$D_{1,n-2,1}$ $D_{1,n-2,2}$ ... $D_{1,n-2,n-2}$ - duritatea liniei (n-2), nivelul 1

...

(*primul nivel are* n-2 *linii și* n-2 *coloane*)

...

$D_{k,1,1}$ - duritatea singurului bloc de pe ultimul nivel

ultimul nivel are o linie și o coloană

Datele de ieșire

Fișierul text PIRAMIDA.OUT care are următoarea structură:

S L - munca depusă și lungimea totală a drumului

h_1 l_1 c_1 - coordonata primului bloc în care se sapă canalul (nivel, linie, coloană)

h_2 l_2 c_2 - coordonata celui de-al doilea bloc în care se sapă canalul (nivel, linie, coloană)

...

h_L l_L c_L - coordonata ultimului bloc în care se sapă canalul (nivel, linie, coloană)

Exemplu:

PIRAMIDA.IN	PIRAMIDA.OUT
5	
1 2 3 4 5	18 6
6 7 8 9 10	0 1 2
11 12 13 14 15	0 2 2
16 17 18 19 20	1 1 1
21 22 23 24 25	1 1 2
1 2 3	1 2 2
4 5 6	2 1 1
7 8 9	
1	

P079807: Rigle Golomb

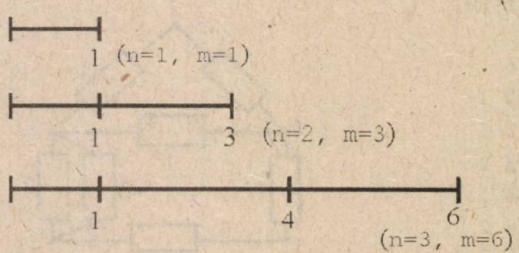
Problemă propusă de conf. dr. Henri Luchian, Universitatea "Al. I. Cuza", Iași

O riglă cu n marcaje se numește riglă Golomb dacă sunt îndeplinite următoarele condiții:

1. cele n marcaje delimitează segmente de lungimi exprimate prin numere naturale; marcajul 0 (originea riglei) nu este inclus între cele n;
2. nu există pe riglă segmente de aceeași lungime (o riglă cu n marcaje are evident n segmente);
3. distanțele ce se măsoară pe riglă, și anume prin unul sau mai multe segmente consecutive, sunt diferite între ele; cu alte cuvinte, nici o distanță nu poate fi măsurată în două moduri diferite.

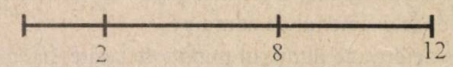
Dacă se adaugă condiției 3) cerința ca toate distanțele cuprinse între 1 și m (lungimea riglei) să poată fi măsurate (evident, în mod unic), atunci avem definiția unei *rigle Golomb perfecte*.

Exemple de rigle Golomb perfecte:



Singurele rigle Golomb perfecte sunt cele trei de mai sus. De aceea, definiția riglei Golomb perfecte trebuie relaxată pentru a o adapta la valori mai mari ale lui n. Anume, condiția 3) plus cerința suplimentară devin: orice distanță cuprinsă între 1 și m poate fi măsurată în cel mult un mod (adică, unele distanțe pot să nu fie măsurabile). Se obține astfel definiția *riglei Golomb bune. Orice riglă Golomb perfectă este și riglă Golomb bună.*

Exemplu de riglă Golomb bună:



Evident, avem o infinitate de rigle Golomb bune cu n marcaje, oricare ar fi n. Analoaga riglei Golomb perfecte cu n marcaje ar fi o riglă Golomb bună cu n marcaje care are lungime minimă.

Dându-se n - numărul de marcaje, să se găsească m - lungimea minimă a unei rigle Golomb bune cu n marcaje.

Date de intrare:

Fișierul GOL.IN conține un număr natural n ($1 \le n \le 16$).

Date de ieșire

Fișierul GOL.OUT va conține n numere naturale reprezentând distanțele față de origine la care trebuie aplicate cele n marcaje pe o riglă Golomb bună de lungime minimă. Ultimul număr va reprezenta deci m, lungimea riglei.

Exemplu:

GOL.IN	GOL.OUT
3	1 4 6

Școala, VIITORUL și INFORMATICA

Clara Ionescu

Valoarea învățământului românesc liceal de informatică pare să fie subiectul unor discuții purtate cu interes nu numai în școală și massmedia ci și în străinătate. Întristată de părerea doamnei profesoare Irina Athanasiu, publicată în interviul din nr. 6 al GInfo am hotărât să stau de vorbă cu dl. prof. Horia Georgescu pentru a adăuga câteva idei și constatări realizate pe parcursul a zeci de ani de preocupări în domeniu.

Știu că sunteți coordonatorul unui program Tempus a cărui temă este strâns legată de învățământul de informatică. Ce amănunte ne puteți da?

Programul *Tempus S_JEP 11168-96* are titlul "*Restructuring of the (re)training of school teachers in Computer Science*". Participă universitățile din Hamburg, Granada, Sheffield, Aalborg, București, Brașov, Cluj, Constanța, Iași, precum și Academia de Studii Economice din București. Sunt cooptate și inspectoratele școlare din centrele universitare românești menționate.

Scopurile principale ale proiectului sunt următoarele:

- crearea de centre regionale puternice pentru activități de perfecționare și reconvertire. Pentru aceasta a fost necesară cunoașterea experienței în domeniu a țărilor vestice participante, echiparea centrelor regionale cu tehnică de calcul modernă și nu în ultimul rând modificarea legislației pentru a sprijini învățământul liceal de informatică și, mai general, a statutului informaticii și a slujitorilor săi.
- perfecționarea în informatică a profesorilor de liceu, în cadrul centrelor regionale create.

În primul an al proiectului (1996/1997), principala realizare structurală a fost modificarea legii învățământului, prin specificarea în art. 161, al. 2 al dreptului absolvenților cursurilor postuniversitare de informatică de a profesa în noua specializare; acest drept este înscris și în *Statutul cadrelor didactice*.

Cursurile postuniversitare au început deja în București, Constanța și Cluj (aici sub formă de învățământ la distanță), la Iași și Brașov. S-a înregistrat o mare afluență de candidați și a fost necesară o selectare a lor.

În al doilea an s-a reușit recunoașterea informaticii ca *știință*. În mod paradoxal, informatica nu figura ca știință ("profil" în accepțiunea oficială), ci ca specializare în cadrul altor profiluri (matematică, inginerie, economie etc.).

Eforturilor de mai mulți ani ai specialiștilor din universitățile din țară li s-au adăugat prezentarea situației din țările vestice (de exemplu în Germania din cele 33 departamente/facultăți de informatică nici una nu funcționează în cadrul unei facultăți de matematică, 5 aparțin unor facultăți de matematică-informatică, 24 sunt de sine stătătoare, iar 4 funcționează în cadrul unor facultăți tehnice).

În primii doi ani ai proiectului, centrele regionale din București, Brașov, Cluj, Constanța și Iași au fost dotate cu câte două laboratoare de câte 10 stații de lucru legate la un server performant. Pentru buna desfășurare a activităților de perfecționare fiecare centru regional a mai primit câte un retroproiector și o placă LCD. În al treilea an se prevede adăugarea unor facilități multimedia.

S-au desfășurat două tipuri de cursuri: pentru perfecționarea profesorilor de informatică și pentru inițierea profesorilor de orice specialitate în *Tehnologia Informației*. Ultimul tip de cursuri a fost cel care s-a bucurat de un mult mai mare succes și va fi cel care va fi continuat în 1998/1999.

Activitatea participanților la proiect a fost inclusă într-o pagină Web, care poate fi consultată la adresele:

`http::/oroles.cs.unibuc.ro/~tempus`
`http://nats-www.informatik.uni-hamburg.de`
`(Project CompTrain)`

precum și în două volume editate în

câte 800 de exemplare și difuzate gratuit la liceele din țară prin intermediul inspectoratelor județene. Volumele conțin trei tipuri de articole: de informare asupra învățământului de informatică din țările vestice, articole având caracter didactic și articole asupra unor tendințe moderne în informatică (acestea pot fi folosite de exemplu pentru lucrările de gradul I).

Ca profesor într-o facultate de specialitate, cum apreciați învățământul de informatică la nivel liceal?

În primul rând țin să subliniez că sunt de acord cu aprecierile făcute în paginile *GInfo* de d-na profesor Irina Athanasiu pe care o admir pentru întreaga sa activitate.

Situația învățământului de informatică la nivel de gimnaziu și liceu (evit termenul de învățământ preuniversitar, pentru că rolul liceului nu este în primul rând cel de a furniza candidați facultăților; în plus, ce ar fi dacă ne-am intitula toți "preacademicieni"?) este foarte dificilă și voi analiza în continuare câteva dintre cauze.

1) *Numărul profesorilor de informatică* (pentru a nu mai vorbi de numărul profesorilor de informatică bine pregătiți) este foarte mic față de cerințe.
   O cauză este salariul foarte mic. Veți spune "dar salariul tuturor profesorilor din licee este mic". Răspunsul este însă nesatisfăcător; dacă cititorul acestor rânduri va avea curiozitatea să răsfoiască volumul Tempus pe 1997/1998, va afla că în Spania salariul unui profesor de informatică este mai mare decât al colegilor de alte specialități, pentru că se ține cont că piața de muncă îi oferă posibilitatea de a câștiga de câteva ori mai mult în afara sistemului de învățământ.
   O altă cauză o constituie complexul de superioritate al facultăților ce scot profesori de informatică, complex transmis și absolvenților. O recentă consecință este hotărârea aberantă a M.E.N. de a bloca accesul inginerilor și economiștilor la susținerea gradelor didactice (din păcate unii au făcut o legătură nedorită și inexistentă cu introducerea cursurilor postuniversitare de specializare). La reuniunea generală Tempus de la Iași în luna iulie am avut surpriza neplăcută de a auzi opinii de tipul "fizicienii nu au ce căuta în informatică". În acest context voi aminti că șeful Departamentului de Informatică al *Universității din Hamburg*, profesorul *Walther v. Hahn* este de formație umanistă, este filolog. Trebuie să ne debarasăm rapid de acest complex, chiar dacă de multe ori noi suntem cei priviți cu superioritate de colegi de la alte discipline. Trebuie să înceteze etichetările din interiorul informaticii (fie că vin de la matematicieni, ingineri sau economiști) și să căutăm un drum comun.
   Poate că acest complex stă și la baza ignorării absolvenților *Colegiilor de informatică*, prin interzicerea accesului lor la modulul pedagogic; în acest mod se pierde posibilitatea de a atrage în gimnaziu profesori calificați.
2) *Dotarea foarte slabă a liceelor în care se predă informatica*. Din dorința (firească) a unor profesori sau/și directori de liceu de a se evidenția, au apărut foarte multe clase de informatică, fără ca dotarea pusă la dispoziția elevilor să fie cea corespunzătoare. Înzestra-

Carte de vizită

Domnul prof. univ. dr. Horia Georgescu este cadru didactic al Facultății de Matematică a Universității București unde a ținut cursuri în domeniul limbajelor de programare, tehnicilor de programare, limbajelor formale, calculului paralel și concurent.

A publicat numeroase articole și cărți de specialitate atât în țară cât și în străinătate.

Este fondatorul *Gazetei de Informatică*, unica publicație de acest gen din lume cu apariție regulată.

În perioada 1990-1994 a pregătit lotul olimpic de informatică aducând țara noastră de pe locul 20 în 1990 pe locul 1 (singurul loc 1 obținut de o echipă a României la olimpiadele internaționale indiferent de disciplină) în 1993 la Mendoza, în Argentina.

A fost inițiatorul *Balcaniadei de Informatică* și a *Olimpiadei de Informatică a Europei Centrale*, organizate în premieră în România.

A condus echipa facultății la concursurile ACM, echipă ce a ocupat primul loc la faza regională (est europeană) în 1995, 1996, 1998 cu drept de participare a României la faza mondială unde s-a obținut o clasare onorabilă (locurile 10-20).

O pasiune recentă: Java.

rea tuturor şcolilor cu tehnică de calcul, aflată în atenţia M.E.N., este o intenţie lăudabilă, dar a cărei punere în practică va întârzia (după părerea mea) cel puţin un deceniu: înzestrarea cu 1-5 calculatoare a unei unităţi şcolare nu va însemna nimic (minimul este o reţea de 10-12 calculatoare), iar simpla prezenţă a unor calculatoare fără a avea la dispoziţie (a elabora) software educaţional nu este suficientă.

3) *Elaborarea unor planuri de învăţământ şi programe analitice* care să ţină cont de:
   - realitatea existentă (valoarea medie a elevilor) şi nu de o idealizare a ei (cei mai dotaţi elevi din ţară);
   - ce urmărim prin introducerea informaticii;
   - dacă putem realiza ceea ce ne propunem.

   Din fericire, la acest capitol situaţia este mai bună. M.E.N. a stabilit ca obligatorie disciplina "*Tehnologia informaţiei*" în licee, pentru toţi elevii. În clasele speciale de matematică-informatică (în conformitate cu discuţia de mai sus sper să devină clase speciale de informatică) se vor studia elemente de informatică. O propunere de programe analitice poate fi găsită în volumul Tempus 1998/1999, întocmită într-un stil tradiţional. Cel mai neclar lucru este ce dorim de la clasele speciale de informatică. Cel mai comod pentru profesorii de liceu este să reproducă fragmente din cursurile urmărite pe băncile facultăţii. Ce va face însă un absolvent de liceu cu aceste informaţii în ipoteza că (voluntar sau nu) nu urmează o facultate de profil? Unde va găsi de lucru cu rudimente de programare? Realitatea arată că în momentul de faţă se cer specialişti cu studii medii nu în programare, ci în depanarea calculatoarelor şi în lucru pe reţele. Cât despre absolvenţii claselor de informatică ce vin în facultăţi, de specialitate, ei de multe ori trebuie reînvăţaţi să gândească (lucru valabil atât pentru universităţile tradiţionale, cât şi pentru cele tehnice sau economice).

4) *Colaborarea continuă dintre profesorii de liceu şi cadrele didactice universitare*. Această colaborare este practic nulă. Deşi ***Legea Învăţământului*** prevede clar că reciclarea profesorilor se face la fiecare 5 ani şi anume în instituţiile de învăţământ superior, în practică nu s-a făcut mai nimic. Din nou este vorba de un amestec de complexe, de data aceasta ale profesorilor de liceu. Ideea este că "noi ştim mai bine ce se întâmplă în licee şi în consecinţă nu avem nevoie de părerea şi amestecul universitarilor". Această părere, cu o largă răspândire, este încurajată de atitudinea M.E.N., care în comisiile de specialitate "acceptă" doar un singur universitar. Este ca şi când am goni din Senatele Universităţilor pe academicieni! Este unul dintre motivele pentru care m-am retras din *Comisia Naţională de Informatică* a M.E.N. Eliminarea rupturii între cadrele didactice din licee şi cele universitare este o condiţie absolut obligatorie pentru a urni lucrurile din loc. Acest lucru este adevărat pentru orice disciplină, dar mai ales pentru informatică, unde lucrurile sunt în permanentă schimbare.

Să revenim la revista noastră, de care v-aţi ocupat de la începuturile ei. Ce ne puteţi spune?

Evident, îmi aduc aminte cu nostalgie de începuturi, când a avea acces la o imprimantă laser însemna a avea o mare şansă, la cele patru drumuri lunare la gară pentru a primi sau trimite cu nasu' formele intermediare ale revistei, şi nu uit că succesul atât cât este al *GInfo* se datorează cel puţin în egală măsură şi Dvs. Nu îmi place însă să zăbovesc prea mult asupra trecutului. Dacă ar fi să vorbim de o constantă în activitatea revistei, aceasta este lipsa de participare a profesorilor de liceu. În 7 ani de activitate, am primit cel mult 20 de materiale de la ei. Din fericire, am fost sprijiniţi de elevi, studenţi, cadre didactice universitare şi informaticieni din afara sistemului de învăţământ. Am căutat împreună să găsim lucrurile cele mai interesante pentru cititori, dar din păcate nu am primit destule reacţii pentru a ne stabili mai bine orientarea, aşa că încă mai bâjbâim...

Oricum, nici în continuare nu vor lipsi probleme propuse şi rezolvate, noutăţi de ultimă oră atât din domeniul software, cât şi cel hardware.

Mai este informatica o ştiinţă de viitor? Cât timp va mai fi valabilă această afirmaţie?

Nu există nici o îndoială că informatica este o ştiinţă de viitor şi că un tânăr ce se specializează în acest domeniu nu va avea grija unui loc de muncă (evident dacă se va pregăti corespunzător). Voi încerca în continuare să argumentez această opinie. În anii 40', *Alan Turing*, (1912-1954) părintele inteligenţei artificiale, formula următoarea întrebare: "*If a computer could think, how would we tell?*" (dacă un calculator ar putea să gândească, cum am putea să ne dăm seama?). Pentru un plus de precizie, ulterior a fost formulat "*testul Turing*" descris în continuare. Se pun întrebări unui om şi unui calculator. Un arbitru analizează răspunsurile şi decide care sunt răspunsurile calculatorului. Până acum, nici un (program de) calculator nu a trecut testul Turing (nu a fost nici o problemă pentru arbitru să discearnă care au fost răspunsurile calculatorului). Drept urmare, premiul de $100,000 a rămas neatribuit. De altfel, se apreciază că în prezent cel mai inteligent program de calculator are inteligenţa unei albine, iar peste 10 ani o va avea pe cea a unei pisici.

Iată deci că mai sunt multe de făcut!.

Vă mulţumesc!

Mașina și bicicleta

Software EDUCAȚIONAL

Mircea Sârbu

În momentul în care se pune tot mai stringent problema informatizării școlilor și a modernizării procesului didactic, este firesc să ne întrebăm la ce vor fi folosite ipoteticele computere ce vor ajunge pe pupitrele elevilor de pe întreg cuprinsul patriei. Cu software educațional, desigur. Da, dar...

În această vară am avut șansa unei întrevederi cu dl. prof. *Andrei Marga*, ministrul Educației Naționale. Cu această ocazie m-am convins, odată în plus, de bunele intenții ale domniei sale și ale colaboratorilor săi. Tot cu această ocazie am purtat o lungă discuție cu dl. *Teodor Gugoiu*, directorul Direcției Informatizare din cadrul ministerului. Printre problemele pe care le-am discutat s-a numărat și cea a softurilor educaționale destinate școlilor. Am aflat cu acest prilej că există o preocupare în această direcție, se vizează editarea unui catalog cu astfel de produse și că, în toamnă, va fi organizată o expoziție la Sala Palatului, cu care ocazie producătorii se vor întâlni cu potențialii lor clienți.

Minunat. Problema care nu era însă lămurită era: cine sunt, de fapt, clienții. În principiu, școlile. Profesorii vor căuta programele pe care le consideră cele mai potrivite pentru materiile pe care le predau și, probabil, școlile le vor achiziționa, spre satisfacția elevilor, profesorilor și, evident, a producătorilor.

Situația este însă puțin mai complicată, chiar dacă trecem peste spinoasa problemă financiară: de unde vor avea școlile bani? În mod normal, aceste programe nu sunt altceva decât *material didactic*, deci nu ar trebui să fie tratate diferit. Dar este îndoielnic faptul că producătorii vor avea forța comercială să penetreze până la potențialii clienți, să vândă suficient și să evite fraudele, astfel încât afacerea să fie suficient de rentabilă pentru a fi susținută și dezvoltată.

În săptămânile următoare am descoperit pe paginile Web ale Ministerului Educației, un text destul de generic privind softul educațional, cu trimitere la o anexă, care însă nu era publicată. În plus, am găsit o listă cu producători de astfel de software. Le-am scris tuturor acestor producători (care aveau adresă de e-mail), cerându-le produsele spre evaluare și prezentare în revistă. Am primit doar două, pe care vi le prezentăm în paginile următoare.

La începutul lunii octombrie am vizitat și expoziția de la Sala Palatului. Concluzia: doar ***două*** firme prezente cu astfel de produse (Softwin și ICE Felix, fiecare cu câte un produs din această categorie). Toți ceilalți producători erau organizații non-profit. Aceasta în condițiile în care softul educațional în alte țări este o industrie înfloritoare.

Concluzia este evidentă. Câtă vreme producătorii nu vor avea șansa să câștige pe această piață, nu vor produce software educațional, oricât de lăudabile ar fi intențiile Ministerului Educației.

V-am rămas dator cu o explicație: de ce "Mașina și bicicleta" ca supra-titlu? Se zice că cineva a întrebat la Radio Erevan: "E adevărat că Ivan Ivanovici a câștigat o mașină?" Răspunsul a fost: "Da... dar nu chiar o mașină, ci doar o bicicletă. Și de fapt nu a câștigat-o, ci i-a fost furată..."

Întrebări

Am adresat aceste întrebări d-lui Teodor Gugoiu, directorul Direcției Informatizare din cadrul Ministerului Educației Naționale. Nu am primit încă răspuns la ele. Mai așteptăm...

1. Care sunt principalele proiecte de informatizare lansate sau pregătite de Ministerul Educației?
2. Rectificarea bugetară intervenită în această toamnă va afecta derularea acestor proiecte?
3. În privința softurilor educaționale, Ministerul Educației are o strategie de stimulare a firmelor producătoare?
4. Se vizează o relaționare a acestor softuri cu programa școlară?

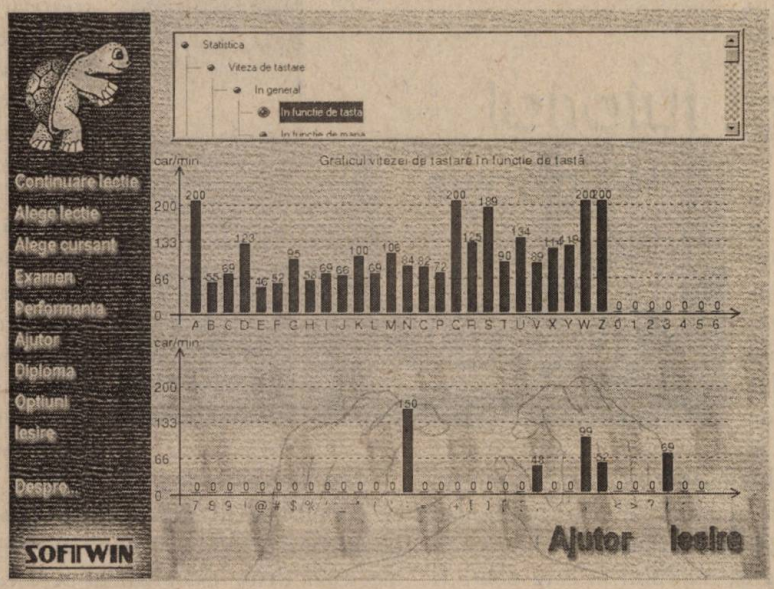
RODACTILO

Cristian Matei

Firma SOFTWIN, le propune celor pentru care calculatorul este sau va fi parte din activitatea curentă, posibilitatea de a tasta cu toate cele 10 degete, prin intermediul aplicației RoDactilo. Aceasta se adresează atât celor care au primul contact cu calculatorul, cât și celor ce au câțiva ani buni de experiență în spate, dar care nu utilizează mai mult de trei, patru degete pentru a scrie un text.

Organizată sub forma unui curs de dactilografie, RoDactilo este prima aplicație românească din acest domeniu. Ea se prezintă sub forma unor lecții, fiecare dintre acestea conținând o serie de exerciții bazate pe o anumită temă. Lecțiile sunt grupate în opt categorii, fiecare categorie terminându-se cu o serie de exerciții recapitulative.

Ca și conținut, găsim trei tipuri de lecții. Primul tip ce cuprinde o lecție introductivă, îi face cunoștință utilizatorului cu tastatura și îi prezintă toate tastele ce conțin literele alfabetului. Cel de-al doilea tip de lecții, îl învață pe utilizator să scrie mai întâi cu un deget, apoi cu două, ajungând progresiv să scrie cu toate cele 10 degete. În sfârșit, cea de-a treia categorie de lecții cuprinde o serie întreagă de texte din domenii diverse, incluzând poezii, fragmente din texte istorice, din astronomie, geografie, glume ș.a.m.d. Conținutul lecțiilor și succesiunea lor se bazează pe metode consacrate de învățare a dactilografiei.



Fiecare exercițiu are două etape, una de învățare și exersare a noilor cunoștințe și una de testare a lucrurilor învățate. Există și un test general, care odată luat vă poate aduce o diplomă de absolvire, eliberară de către RoDactilo și care garantează că cursantul știe să tasteze cu peste 200 de caractere / sec, cu un număr minim de erori.

Textele de la fiecare exercițiu îi sunt afișate utilizatorului, iar la cererea acestuia îi pot fi dictate, existând trei viteze diferite de dictare. De asemenea, pentru a veni în întâmpinarea diferitelor tastaturi de PC-uri, RoDactilo a fost înzestrată cu capacitatea de a se adapta pentru tastaturi românești, americane, franceze, engleze, italiene și germane. În cazul literelor cu diacritice, pot fi specificate tastele sau combinațiile de taste utilizate de fiecare utilizator în parte, pe calculatorul său.

Pentru a ține o evidență a performanțelor realizate, RoDactilo poate genera o serie de statistici complexe legate de viteza de tastare, pe diferite criterii. Date despre viteza de tastare și procentul erorilor, sunt afișate chiar și în timpul exercițiilor și la sfârșitul fiecăruia dintre ele.

RoDactilo a fost proiectată pentru a ține evidența pentru mai mulți utilizatori, reținând pentru fiecare dintre aceștia, la ce lecție a ajuns și care sunt performanțele individuale ale fiecăruia.

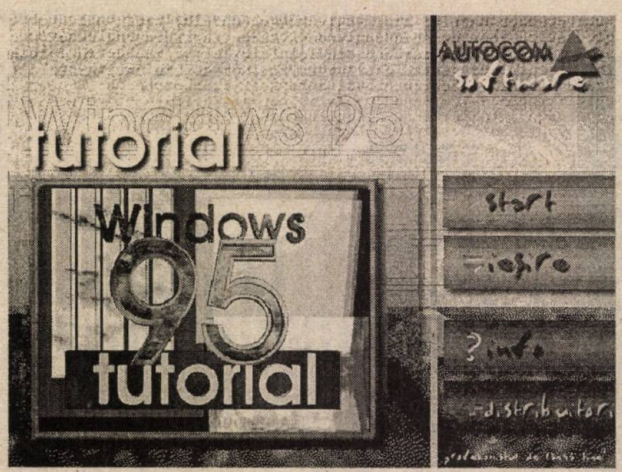
Pentru mai multe informații sau pentru a achiziționa aplicația RoDactilo, puteți apela la firma SOFTWIN din București, tel 01-2305026 sau via e-mail la office@softwin.ro.

WINDOWS '95

Tutorial

Cristian Matei

Puțini utilizatori ai sistemului de operare Windows, cunosc toate facilitățile oferite de către acesta. Pentru a veni în sprijinul acestora, dar și a celor ce pătrund pentru prima dată în lumea oferită de Windows 95, firma Autocom Software a realizat o prezentare, numită Windows 95 Tutorial. Aceasta tratează toate aspectele importante ale sistemului de operare Windows 95, atât în mod teoretic, cât și în mod practic, punând utilizatorul să învețe și să aplice lucrurile învățate, în același timp, fiind îndrumat pe cale audio, ca și cum s-ar afla într-o adevărată sală de clasă, în fața unui calculator special destinat acestui lucru.



Lecțiile încep cu prezentarea modului în care este instalat sistemul de operare Windows 95, urmată de explicarea termenilor noi ce apar în acesta. Celor obișnuiți cu sistemul de operare Windows 3.1, le sunt prezentate schimbările apărute odată cu migrarea spre noul sistem de operare, iar celor noi în domeniu, le sunt prezentate taskurile și lucrul cu acestea. Urmează tratarea modului în care poate fi configurată interfața utilizator, pentru a face plăcut și a reduce la minim efortul depus de utilizatorul sistemului de operare Windows 95.

Principalelor aplicații prezente cu Windows 95, le sunt afectate lecții separate, pentru a obișnui utilizatorul cu anumite concepte comune din Windows 95. Astfel, odată cu Windows Explorer, este prezentat lucrul cu fișierele, odată cu Microsoft Exchange, lucrul cu mesaje, cu WordPad, lucrul cu texte și cu obiectele generate de diferite aplicații, iar odată cu Paint, modul în care se lucrează cu culorile și imaginile grafice. La acestea, se adaugă și câteva lecții mai scurte destinate celorlalte aplicații din Windows 95.

Printre lecții, se regăsesc și unele destinate modului în care sunt instalate imprimantele și modemurile, modului în care se face conectarea calculatorului într-o rețea sau cu un laptop și lecții destinate creșterii performanțelor sistemului de operare Windows 95. Tot în aceste lecții sunt prezentate și modurile de utilizare al acestor facilități oferite de către Windows 95.

Tot în cadrul prezentării, sunt incluse și informații privind câteva firme românești furnizoare de produse software și hardware, legate de sistemul de operare Windows 95.

Deoarece Windows 95 Tutorial rulează direct de pe CD-ROM-ul utilizat ca și suport, sub sistemele de operare Windows 95 sau Windows 3.1, acesta nu afectează deloc calculatorul pe care a fost lansat în execuție, toate operațiile executate de către cursant fiind simulate. *(Crystian)*

Pentru mai multe informații sau pentru a comanda CD-ROM-ul cu prezentarea Windows 95 apelați la firma Autocom Software din Timișoara, tel. 056>201460.

Arbori STEINER

Horia Giurgiu

Un titlu din teoria grafurilor...
Sunteți tentați să "dați pagina" oarecum îngroziți de plictiseala unor expuneri matematice? De data aceasta să n-o faceți! Arborii Steiner au o aplicabilitate extrem de frumoasă în proiectarea... microprocesoarelor.

Distanța Manhattan

Atunci când doreau să realizeze un cablaj la o schemă nouă, cei pasionați de electronică s-au întâlnit de multe ori cu problema *conectării grupurilor de pini.*

Această problemă se poate transpune în limbajul teoriei grafurilor astfel: *fiecare pin al unui circuit reprezintă un nod, iar toți pinii care se conectează între ei formează un subgraf complet (fiecare nod e conectat cu toate celelalte).*

Astfel problema inițială se reduce la următoarea: dându-se un graf format din mai multe subgrafuri complete, se cere să se determine subgrafurile conexe aferente, adică pentru fiecare subgraf se caută un arbore parțial asociat. Practic un arbore parțial conține toate nodurile grafului/subgrafului inițial și o parte din muchii (mai exact doar n-1, unde n este numărul de noduri).

În general se caută arborii parțiali care au o anumită proprietate suplimentară, ca de exemplu *suma lungimilor tuturor muchiilor arborelui să fie minimă.* În acest caz subgraful se numește *arbore parțial minim.* Pentru determinarea acestuia există algoritmi bine puși la punct care generează soluția optimă într-un timp polinomial.

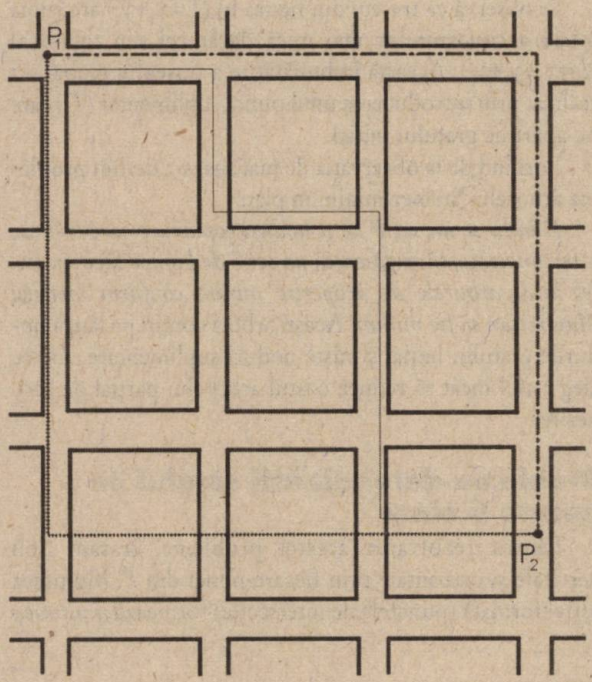
Tot în cadrul electronicii, pentru realizarea anumitor circuite digitale și mai ales în proiectarea microprocesoarelor, s-a mai impus o restricție: *toate liniile de conexiune dintre puncte să fie formate doar din segmente orizontale și verticale.* În acest caz, pentru calcularea distanței dintre două puncte, adică a lungimii unei muchii, nu se mai folosește formula clasică pentru distanță (metrica euclidiană):

$$d(N_1, N_2) = \sqrt{(x_1-x_2)^2 + (y_1-y_2)^2}$$

ci așa-numita *distanță Manhattan* (metrica Manhattan):

$$d(N_1, N_2) = |x_1-x_2| + |y_1-y_2|$$

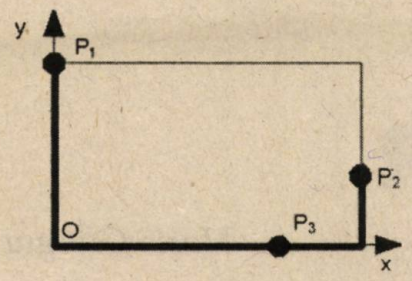
Denumirea de *distanță Manhattan* vine de la următoarea situație: în Manhattan străzile au fost realizate astfel:



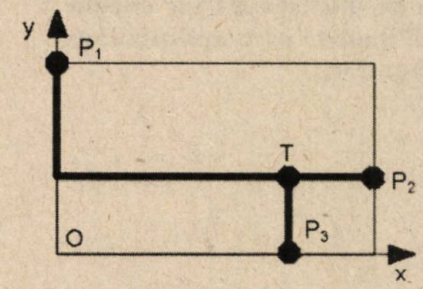
Străzile din Manhattan

După cum se poate vedea în desenul de mai sus cele trei drumuri figurate au aceeași lungime. Practic, între două puncte care nu se află pe aceeași linie orizontală sau verticală, există o infinitate de drumuri care conectează cele două puncte și toate au aceeași valoare minimă.

Dacă studiem următoarea situație în care trebuie să conectăm trei puncte:



a)



b)

Compararea traseelor

Se observă că traseul din figura b) ($L=x_2+y_1$) are suma totală a conexiunilor mai mică decât cel din figura a) ($L=x_2+y_1+y_2$). Această îmbunătățire a traseului global s-a realizat prin introducerea unui punct suplimentar (T), care nu aparține grafului inițial.

Pornind de la observația de mai sus, s-a definit problema arborelui Steiner minim în plan:

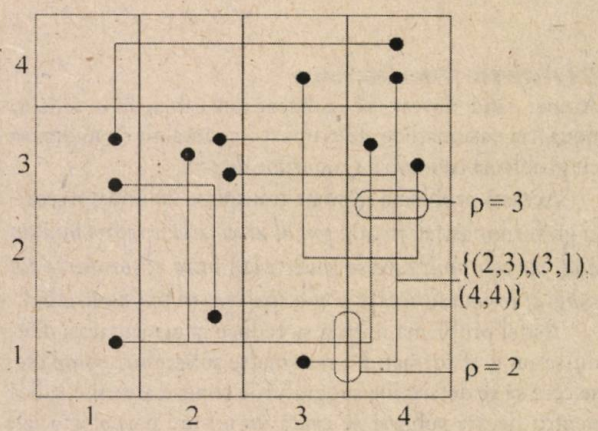
Dându-se un set P de n noduri coplanare care trebuie interconectate, să se găsească un set S de puncte Steiner astfel încât arborele de acoperire minim conform metricii Manhattan să fie minim. Acești arbori conțin pe lângă nodurile grafului inițial și niște noduri suplimentare care se aleg astfel încât să reducă costul arborelui parțial de acoperire.

Problema determinării setului de puncte Steiner

Pentru rezolvarea acestei probleme, trasăm linii verticale și orizontale prin fiecare punct din P. Mulțimea astfel formată (punctele de intersecție) formează *mulțimea de puncte Steiner candidat* S_c. Se poate demonstra că $S \subset S_c$.

S-a demonstrat că problema deteminării setului de puncte Steiner S este NP-completă. Din această cauză, implementările actuale folosesc algoritmi euristici care furnizează o soluție apropiată de optim, dar care au o complexitate polinomială.

Revenind la problema de la care am pornit de a conecta grupuri de pini astfel încât lungimea totală a traseelor să fie minimă, s-a observat că aceasta implică o minimizare a sumei costurilor subarborilor Steiner. O soluție ar fi determinarea subarborilor Steiner pentru fiecare componentă în parte, dar această metodă duce la o aglomerare a traseelor care nu poate fi realizată în practică. Limitarea fizică principală a construcției de cablaje este densitatea maximă limitată a traseelor. Pentru transpunerea ei în spațiul matematic al grafurilor s-a introdus o grilă de control a densității. Aceasta este formată din pătrate pentru care s-a definit numărul maxim de trasee care pot traversa o latură oarecare a grilei.



Grila de control a densităților

Și în acest caz problema este NP-completă. În practică se preferă construirea acestor arbori Steiner cu ajutorul unor algoritmi euristici de rutare (complexitate polinomială) de tip *greedy* care asigură soluții apropiate de optim.

---

dl. Horia Giurgiu este asistent la Facultatea de Calculatoare, Universitatea Tehnică din Cluj, și poate fi contactat la adresa de e-mail: Horia.Giurgiu@cs.utcluj.ro

Problema FLUXULUI MAXIM

Eleonor Ciurea

Unul din subiectele "tari" care se studiază și în liceu... Problema fluxului maxim se ascunde - subtil - în multe probleme de rezolvat cum este și cea din acest articol care propune stabilirea unui consiliu de guvernare a unui oraș astfel încât... Amănuntele le puteți afla din cele ce urmează.

1. Notații și ipoteze

Considerăm un graf orientat $G = (N, A)$ pentru care N este mulțimea nodurilor, $|N|=n$, A este mulțimea arcelor, $|A|=m$, și $c : A \to R_+$ este funcția care asociază fiecărui arc $(x, y) \in A$ care asociază un număr real nenegativ $c(x, y)$ numit capacitatea arcului (x, y).

Problema fluxului maxim constă în a determina funcția $f : A \to R_+$ care verifică

$$\sum_y f(x, y) - \sum_y f(y, x) = \begin{cases} v, & x = s, \\ 0, & x \neq s, \bar{s}, \\ -v, & x = \bar{s}, \end{cases} \qquad (1)$$

$$0 \leq f(x, y) \leq c(x, y), (x, y) \in A, \qquad (2)$$

maximizează v. (3)

Constrângerile (1) se numesc *constrângeri de echilibrare*, iar constrângerile (2) se numesc *constrângeri de mărginire*. Valoarea corespunzătoare variabilei v este *valoarea fluxului*, nodul s se numește *nod origine* sau *sursă*, nodul $\bar{s}$ se numește *nod destinație* sau *stoc*, celelalte noduri $x \neq s$, $\bar{s}$ se numesc *noduri intermediare*.

Un graf orientat $G = (N, A)$ pe care s-a definit *funcția capacitate* c se numește *rețea* pe care o notăm $R = (G, c)$. În continuare considerăm:

$$C = \max(c(x, y) | (x, y) \in A).$$

Problema fluxului maxim o supunem următoarelor ipoteze.

Ipoteze:

Rețeaua este orientată.

Toate capacitățile sunt întregi nenegative.

Rețeaua nu conține nici un drum de la nodul sursă s la nodul stoc $\bar{s}$ alcătuit numai din arce având capacități infinite.

Dacă un arc $(x, y) \in A$, atunci arcul $(y, x) \in A$.

Rețeaua nu conține arce paralele (adică două sau mai multe arce cu același nod inițial și final).

2. Aplicații

Problema fluxului maxim și problema tăieturii minime apar într-o varietate de domenii. În continuare vom descrie două aplicații.

Problema ofertei și cererii

Presupunem că există anumite centre de ofertă cu același produs și anumite centre de cerere ale acestui produs. Cunoaștem oferta produsului, cererea produsului și cantitatea maximă de produs care se poate transporta pe fiecare rută. Dorim să cunoaștem dacă se pot satisface toate cererile prin utilizarea ofertelor. Problema se poate modela ca o problemă a *fluxului admisibil* care cere să identificăm un flux f într-o rețea $R = (G, c)$ satisfăcând constrângerile:

$$\sum_y f(x, y) - \sum_y f(y, x) = v(x), \; x \in N \qquad (4)$$

$$0 \leq f(x, y) \leq c(x, y), \quad (x, y) \in A \qquad (5)$$

unde $\sum_N v(x) = 0$.

Se poate rezolva problema fluxului admisibil în rețeaua $R = (G, c)$ prin rezolvarea unei probleme de flux maxim în rețeaua transformată:

$R' = (G', c')$ cu $G' = (N', A')$, $c' : A' \to R_+$ unde

$N' = N_1 \cup N_2 \cup N_3$,

$N_1 = \{s'\}$, $N_2 = N$, $N_3 = \{\bar{s}\}$,

$A' = A_1 \cup A_2 \cup A_3$

$A_1 = \{(s', x) / v(x) > 0\}$, $A_2 = A$, $A_3 = \{(x, \bar{s}') / v(x) < 0\}$,

$c'(s', x) = v(x)$ $\quad (s', x) \in A_1$,

$c'(x, y) = c(x, y)$ $\quad (x, y) \in A_2$,

$c'(x, \bar{s}') = -v(x)$ $\quad (x, \bar{s}') \in A_3$.

Nodul s' se numește *supersursă*, iar nodul $\bar{s}'$ se numește *superstoc*.

Arcul $(s', x) \in A_1$ se numește *arc sursă*, iar arcul $(x, \bar{s}')$ se numește *arc stoc*.

Proprietate:

În rețeaua originală R există un flux admisibil dacă și numai dacă în rețeaua transformată R' există un flux maxim care saturează toate arcele sursă și stoc.

Problema reprezentărilor

Un oraș are p cluburi, q locuitori și r partide politice. Fiecare locuitor este membru la cel puțin un club și poate să facă parte doar dintr-un singur partid. Fiecare club trebuie să-și desemneze un membru care să-l reprezinte la consiliul de guvernare al orașului astfel încât numărul membrilor din consiliu care fac parte din partidul p_k este cel mult u_k. Dacă există un astfel de consiliu, spunem că acesta este un *consiliu echilibrat*.

Formulăm această problemă ca o problemă de flux maxim într-o rețea $R = (G, c)$ cu

$G = (N, A), c: A \to R_+$, unde:

$N = N_1 \cup N_2 \cup N_3 \cup N_4$, - mulțimea nodului sursă și nodului stoc

$N_1 = \{s, \bar{s}\}$

$N_2 = \{c_1, ..., c_p\}$ - mulțimea nodurilor care reprezintă cluburile

$N_3 = \{l_1, ..., l_q\}$ - mulțimea nodurilor care reprezintă locuitorii

$N_4 = \{p_1, ..., p_r\}$ - mulțimea nodurilor care reprezintă partidele

Mulțimea arcelor este $A = A_1 \cup A_2 \cup A_3 \cup A_4$, unde:

$A_1 = \{(s_1, c_i) | c_i \in N_2\}$

$A_2 = \{(c_i, l_j) | c_i \in N_2, l_j \in N_3$ și l_j este membru la $c_i\}$

$A_3 = \{(l_j, p_k) | l_j \in N_3, p_k \in N_3$, și l_j face parte din $p_k\}$

$A_4 = \{(p_k, \bar{s}) | p_k \in N_4, p_k \in N_4\}$

Funcția capacitate este $c(x,y) = 1$ dacă (x,y) este din A_1 sau A_2 sau A_3 și $c(p_k, \bar{s}) = u_K$, $(p_k, \bar{s}) \in A_4$.

Evident, dacă valoarea fluxului maxim este p, atunci orașul are un consiliu echilibrat, altfel consiliul nu este echilibrat. Considerăm exemplul prezentat în figura 1 pentru $p = 4$, $q = 7$, $r = 3$.

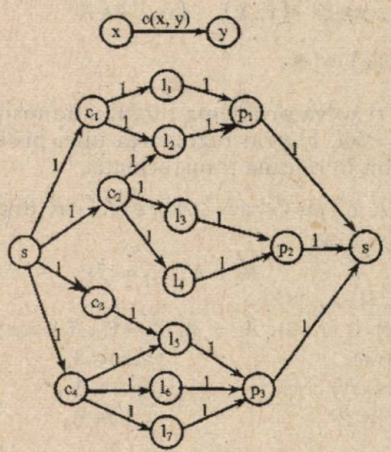


Fig. 1 Problema reprezentanților

3. Fluxuri și tăieturi

În implementarea oricărui algoritm pentru fluxul maxim avem opțiunea să lucrăm fie pe rețeaua originală cu fluxurile $f(x,y)$, fie pe o rețea transformată, numită *rețea reziduală*, cu capacitățile reziduale $r(x,y)$. În acest ultim caz, când algoritmul se termină, trebuie să calculăm fluxurile $f(x,y)$.

Deoarece în ultimul timp, în literatura de specialitate se utilizează rețeaua reziduală, vom prezenta rezultatele și algoritmii pentru fluxul maxim pe această rețea.

Fiind dat un flux f, capacitatea reziduală $r(x, y)$ a oricărui arc $(x, y) \in A$ este fluxul adițional maxim care poate fi trimis de la nodul x la nodul y utilizând arcele (x, y) și (y, x). Capacitatea reziduală $r(x, y)$are două componente:

$c(x, y)$-$f(x, y)$ reprezintă capacitatea nefolosită a arcului (x, y);

$f(y, x)$ reprezintă fluxul pe arcul (y, x) și pe care îl putem micșora pentru a crește fluxul de la nodul x la nodul y.

În consecință $r(x, y) = c(x, y) - f(x, y) + f(y, x)$. Rețeaua reziduală $R(f)$ este formată numai din arcele cu capacitățile reziduale pozitive, adică din arcele (x, y) cu $r(x, y) > 0$. În figura 2b se prezintă rețeaua reziduală $R(f)$ corespunzătoare fluxului f din rețeaua prezentată în figura 2a.

Dacă X și Y sunt două submulțimi ale mulțimii nodurilor N, nu în mod necesar disjuncte, definim mulțimea de arce:

$$(X, Y) = \{(x, y) \mid (x, y) \in A, x \in X, y \in Y)\}$$

Pentru orice funcție $g: N \times N \to R_+$ definim

$$g(X, Y) = \sum_{(X,Y)} g(x, y) \cdot$$

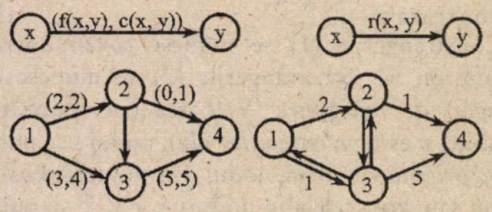


Fig. 2 Rețeaua originală și rețeaua reziduală

Dacă $X=(x)$ sau $Y=(y)$ notăm $g(X, Y)$ prin $g(x,Y)$, respectiv $g(X, y)$.

Cu aceste notații constrângerile de echilibrare (1) se scriu:

$$f(x,N) - f(N,x) = \begin{cases} v, x = s, \\ 0, x \neq s, \bar{s}, \\ -v, x. \end{cases} \qquad (1')$$

Fie $R=(G, c)$ o rețea și $X \subset N$, $\bar{X} = N - X$ cu X, $\bar{X}$ nevide.

O *tăietură* este mulțimea de arce $[X, X] = (X, X) \cup (X, X)$, unde $(X, \bar{X})$ reprezintă mulțimea arcelor directe și $(\bar{X}, X)$ reprezintă mulțimea arcelor

inverse ale acestei tăieturi. Notăm că ordinea în care mulțimile $X, \overline{X}$ sunt înregistrate în $[X,\overline{X}]$ este importantă; schimbarea lor schimbă mulțimea arcelor directe și mulțimea arcelor inverse în tăietură. O tăietură $[X,\overline{X}]$ cu $s \in X$ și $\overline{s} \in \overline{X}$ se numește *tăietură sursă-stoc* (tăietură $s-\overline{s}$). Capacitatea $c[X,\overline{X}]$ a tăieturii $s-\overline{s}$ $[X,\overline{X}]$ este definită a fi $c[X,\overline{X}] = c(X,\overline{X})$. O tăietură $s-\overline{s}$ a cărei capacitate este minimă printre toate tăieturile $s-\overline{s}$ se numește *tăietură $s-\overline{s}$ minimă*. Capacitatea reziduală $r[X,\overline{X}]$ a tăieturii $s-\overline{s}$ $[X,\overline{X}]$ este definită a fi $r[X,\overline{X}] = r(X,\overline{X})$. Dacă f este un flux și $[X,\overline{X}]$ o tăietură $s-\overline{s}$, atunci fluxul net peste această tăietură este $f[X,\overline{X}] = f(X,\overline{X}) - f(\overline{X},\overline{X})$.

Teoreme:

1. *În rețeaua $R = (G,c)$ dacă f este un flux de valoare v, $[X,\overline{X}]$ o tăietură $s-\overline{s}$, atunci f verifică relațiile:* $v = f[X,\overline{X}] \le c[X,\overline{X}]$.
2. *În rețeaua $R = (G, c)$ dacă f este un flux de valoare v, $[X,\overline{X}]$ o tăietură $s-\overline{s}$ și f' un flux de valoare v' cu $v' \ge v$ atunci:* $v = f[X,\overline{X}] \le c[X,\overline{X}]$.

4. Algoritmul generic

În rețeaua $R = (G, c)$, fie f un flux de valoare v, L un lanț de la nodul sursă s la nodul stoc $\overline{s}$ cu L^+ mulțimea arcelor directe și L^- mulțimea arcelor inverse. Un lanț L de la nodul sursă s la nodul stoc $\overline{s}$ se numește lanț de mărire a fluxului (*LMF*) în raport cu fluxul f dacă $f(x, y) < c(x, y)$ pentru $(x, y) \in L^+$ și $f(y, x) > 0$ pentru $(y, x) \in L^-$.

Un *LMFL* în raport cu fluxul f în rețeaua originală R corespunde la un drum D' de mărire a fluxului (*DMF*) în rețeaua reziduală $R(f)$ și viceversa. Definim capacitatea reziduală a unui *LMFL* în modul următor:

$r(L^+) = \min\{(x, y) - f(x, y) / (x, y) \in L^+\}$, $r(L^-) = \min\{f(y, x) / (y, x) \in L^-\}$, $r(L) = \min\{r(L^+), r(L^-)\}$.

Evident că $r(L) > 0$. Atunci se poate face mărirea de flux:

$f'(x, y) = f(x, y) + r(L)$, $(x, y) \in L^+$,

$f'(x, y) = f(y, x) - r(L)$, $(y, x) \in L^-$, $f'(x, y) = f(x, y)$, $(x, y) \notin L$ și fluxul f' are valoarea $v' = v + r(L)$.

Cât timp rețeaua reziduală $R(f)$ conține un *DMF*, putem trimite flux de la nodul sursă s la nodul stoc $\overline{s}$. Algoritmul generic este bazat pe această proprietate.

Ilustrăm algoritmul generic pe rețelele din figura 3. Inițial $f = 0$. Rețeaua reziduală este rețeaua originală din figura 3a. Presupunem că algoritmul generic selectează DMFD'=(1,3,4) cu

$r(D') = min\{r(1,3), r(3,4)\} = min\{4,5\} = 4$.

Mărirea de flux reduce $r(1, 3) = 4$ la 0, astfel se elimină arcul (1,3) din rețeaua reziduală, se crește $r(3,1)$ de la 0 la 4, astfel se adaugă arcul (3,1). De asemenea, mărirea de flux descrește $r(3,4)$ de la 5 la 1, astfel în noua rețea reziduală $r(3,4) = 1$ și $r(4,3) = 4$.

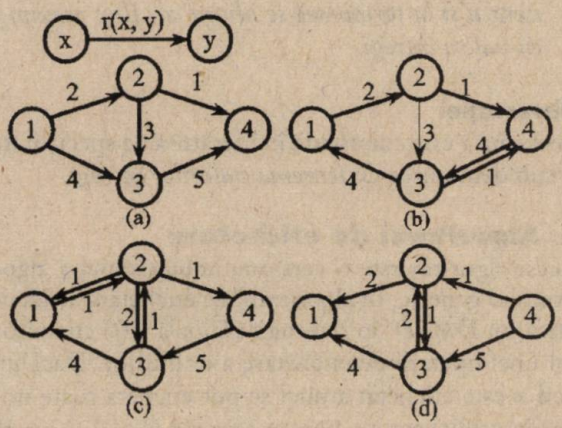


Fig. 3 Ilustrarea algoritmului generic

Figura 3b arată rețeaua reziduală după prima iterație. În a doua iterație presupunem că algoritmul generic selectează *DMFD'* = (1, 2, 3, 4) cu $r(D') = \min\{2, 3, 1\} = 1$. Mărirea de flux conduce la rețeaua reziduală arătată în figura 3c. În a treia iterație algoritmul generic selectează unicul *DMFD'*=(1,2,4) cu $g(D')=\min\{1,1\}=1$. Mărirea de flux conduce la rețeaua reziduală arătată în figura 3d care nu mai conține *DMF* și algoritmul se termină.

```
Program ALGORITM GENERIC;
Begin
  f:=f0;
  while R(f) conține un DMF do
  begin
    se identifică un DMF D';
    r(D'):=min{r(x, y)/(x, y) ∈ D'}
    se execută mărirea de flux;
    se actualizează R(f)
  end
End.
```

Să vedem cum putem determina fluxurile $f(x,y)$ la terminarea algoritmului. În general, mai multe combinații ale lui $f(x,y)$ și $f(y,x)$ corespund pentru a verifica relația $r(x,y) = c(x,y) - f(x,y) + f(y,x)$ și constrângerile (2). Pentru o alegere scriem

$r(x,y) = c(x,y) - f(x,y) + f(y,x)$ pentru ca

$c(x,y) - r(x,y) = f(x,y) - f(y,x)$.

Acum dacă $c(x,y) - r(x,y) \ge 0$, atunci $f(x,y) = c(x,y) - r(x,y)$ și $f(x,y) = 0$, altfel stabilim $f(x,y) = 0$ și $f(y,x) = r(x,y) - c(x,y)$.

Considerând exemplul de mai sus se stabilesc: $c(1,2) - r(1,2) = 2 - 0 = 2$, deci $f(1,2) = 2$. Analog $f(1,3) = 4$, $f(2,3) = 1$, $f(2,4) = 5$, $f(3,4) = 1$.

Teoremă

(3) *În rețeaua $R = (G,c)$ dacă capacitatea c și fluxul admisibil inițial f_0 sunt cu valori întregi atunci al-*

goritmul generic converge într-un număr finit de iterații și la terminare se obține un flux maxim f cu valori întregi.

Observație:
Teorema 3 este cunoscută în literatura de specialitate și sub denumirea de *teorema valorilor întregi.*

5. Algoritmul de etichetare

Acest algoritm este o versiune îmbunătățită a algoritmului generic. În algoritmul de etichetare se identifică un *DMFD'* în rețeaua reziduală *R(f)* cu ajutorul unei operații de etichetare a nodurilor. Dacă un nod *x* este etichetat atunci se pot eticheta toate nodurile neetichetate *y* pentru care $r(x,y) > 0$. Această operație de etichetare se numește *analizarea nodului etichetat x*. Algoritmul de etichetare se datorează lui Ford și Fulkerson (1956), de aceea este cunoscut și sub numele de ***algoritmul Ford-Fulkerson***. În timpul execuției algoritmului de etichetare un nod *x* poate fi etichetat sau neetichetat. Un nod *x* este etichetat dacă algoritmul, prin operația de etichetare, a identificat un drum *D'* de la nodul sursă *s* la nodul *x* în rețeaua reziduală, altfel nodul *x* este neetichetat. Un nod etichetat *x* este analizat dacă sunt etichetate toate nodurile *y* pentru care $r(x,y) > 0$, altfel nodul *x* este neanalizat. În algoritmul de etichetare utilizăm lista nodurilor etichetate și neanalizate *EN* și vectorul predecesor notat *P*.

```
Program ALGORITM ETICHETARE;
Begin
  f:=f0;
  se etichetează nodul s̄;
  while s̄ este etichetat do
  begin
    se elimină toate etichetele;
    se inițializează P(y):=0 pentru fiecare y ∈N;
    se etichetează nodul s;
    se stabilește EN:={s};
    while EN ≠ ϕ și s̄ este neetichetat do
    begin
      se scoate un nod x din EN;
      for fiecare arc (x,y) din R(f) do
        if r(x,y) > 0 și y este neetichetat
        then
        begin
          se stabilește P(y):=x;
          se etichetează y;
          se adaugă y la EN
        end
    end;
    if s̄ este etichetat then MARIRE
  end
End.
Procedura MARIRE;
begin
  se determină DMF D' cu vectorul predecesor P;
  r(D') = min {r(x,y) / (x,y) ∈ D'};
  se efectuează mărirea de flux de-a lungul lui D';
  se actualizează R(f)
end;
```

Teoreme:

(4) *În rețeaua R = (G, c) valoarea fluxului maxim de la nodul sursă s la nodul stoc $\bar{s}$ este egală cu capacitatea tăieturii s-$\bar{s}$ minime.*

(5) *La terminarea algoritmului de etichetare fluxul obținut este un flux maxim.*

(6) *În rețeaua r = (G, c) un flux f^* este un flux maxim dacă și numai dacă rețeaua reziduală $R(f^*)$ nu conține DMF.*

În literatura de specialitate se arată că, dacă se dă rețeaua originală *R = (G, c)*, atunci se poate construi rețeaua transformată $\tilde{R} = (\tilde{G}, \tilde{c})$ astfel încât comportarea algoritmului generic pe rețeaua reziduală *R(f)* este aceeași cu comportarea algoritmului de etichetare pe rețeaua reziduală $\tilde{G}, \tilde{c}$. De aceea, rezultă următoarele concluzii:

c_1) Dacă capacitatea și fluxul inițial sunt cu valori întregi, atunci aplicarea algoritmului de etichetare se termină cu un flux maxim după un număr finit de iterații.

c_2) Dacă capacitățile sunt iraționale, atunci aplicarea algoritmului de etichetare poate produce o secvență infinită de fluxuri ale căror valori converg la o valoare strict mai mică decât valoarea fluxului maxim.

c_3) Complexitatea algoritmului de etichetare este exponențială în raport cu mărimea problemei.

Teoremă

(7) *Dacă capacitatea și fluxul inițial sunt cu valori întregi atunci algoritmul de etichetare are complexitatea* $\mathbf{O}(m\ n\ C) = \mathbf{O}(n3\ C)$, *unde* $C = max\ \{c(x, y) / (x, y) \in A\}$.

Observații:
Deoarece complexitatea algoritmului de etichetare depinde de *C* se spune că el are o complexitate semipolinomială.

Pentru rezolvarea problemei fluxului maxim există multi algoritmi cu complexitate polinomială care se bazează fie pe *DMF* din rețeaua reziduală fie pe noțiunea de preflux.

Domnul Eleonor Ciurea este profesor la Universitatea "Transilvania" din Brașov și poate fi contactat prin e-mail la adresa: ciurea@unitbv.ro

INDEXARE cu "hărți de BIȚI"

Ion Cozac

Metodele de indexare reprezintă un domeniu cu implicații majore în performanțele aplicațiilor care prelucrează volume mari de date. Cunoașterea lor ne poate ajuta să alegem metoda potrivită în funcție de situația concretă. Pe de altă parte, metodele de indexare sunt un câmp de aplicabilitate pentru cunoștințele de algoritmică și abilitățile de programare. Și - de ce nu - pentru imaginație...

Utilizarea acestei tehnici de indexare conduce la un acces mai rapid la o bază de date și la economii importante de spațiu ocupat pe disc. Indecșii bazați pe hărți de biți (*bitmapped indexes*) sunt utili mai ales pentru mediile care prelucrează depozite de date (*data warehouses*), unde interogările ad-hoc sunt mult mai frecvente decât actualizările. Atunci când se utilizează împreună cu alte tehnici convenționale de indexare, cum sunt cele bazate pe B-arbori, indecșii bitmap conferă avantaje deosebite prin îmbunătățirea semnificativă a performanțelor în cazul accesului la depozite de date.

Indecși bitmap versus B-arbori

În articolul din numărul trecut am prezentat o tehnică de indexare bazată pe B-arbori. Aceasta este utilă în cazul în care câmpul ales drept cheie ia valori într-un domeniu foarte mare (având cardinalitate mare), cum este de exemplu *Nume-client* sau *Numar-telefon*. Un index bitmap este mult mai indicat pentru câmpuri care pot lua puține valori distincte.

Exemple:
Sex(B,F), Stare-Civila(N,C,D,V).

Atunci când se creează un index bitmap pentru un câmp, se rezervă atâția biți câte valori distincte poate lua atributul respectiv. Fiecare bit din șirul de biți este asociat unei valori din domeniu, și o valoare 1 sau 0 semnifică faptul că o înregistrare din tabelă are valoarea câmpului egală sau nu cu valoarea asociată poziției respective a bitului în cadrul șirului. Un exemplu de fișier indexat folosind această tehnică este dat în primele două tabele. Câmpurile *Stare-Civila*, *Sex* și *Zona* pot lua foarte puține valori, astfel că ele sunt candidați foarte buni pentru indecși bitmap. Câmpul *Cod-Client* ia valori într-un domeniu care poate avea o cardinalitate foarte mare, așadar pentru acest câmp trebuie folosită o indexare cu ajutorul B-arborilor.

Schița unei implementări

Prima propunere încearcă să obțină un compromis rezonabil între complexitatea operațiilor de actualizare (adăugare, modificare, ștergere) și a celor de căutare (folosind operații logice pe biți).

Fiecare înregistrare din fișierul de date conține:

- valorile tuturor câmpurilor care nu apar în indexul bitmap;
- numărul de ordine (la inserare) al acesteia, începând cu numărul 0 pentru prima înregistrare inserată.

Înregistrarea corespondentă din fișierul bitmap conține următoarele informații:

- adresa relativă (față de începutul fișierului) a înregistrării corespondente din fișierul de date; dacă fișierul de date are înregistrări de lungime variabilă, atunci se memorează și lungimea acesteia;
- un bit pe post de indicator de ștergere, care are valoarea 0 dacă înregistrarea corespondentă este validă, și 1 dacă a fost ștearsă;
- șirul de biți corespunzător valorilor câmpurilor care apar în indexul bitmap.

Pentru simplificarea prelucrării, șirul de biți se completează până la cel mai apropiat multiplu de 8, pentru a putea prelucra în orice moment octeți întregi.

Cu aceste precizări operațiile de actualizare decurg astfel:

Depozite de date

Aproape toate întreprinderile moderne folosesc calculatoarele pentru activitățile de gestiune economică. Aceste activități au drept scop să reflecte în mod *operativ* situația *actuală* a întreprinderii, din diferite puncte de vedere. În acest scop se folosesc aplicații economice (*business applications*) pentru contabilitate, evidența stocurilor, amortizarea mijloacelor fixe etc.
Această categorie de aplicații formează ceea ce se numește "sistemul informatic operativ" al întreprinderii. Caracteristic acestui sistem este faptul că se bazează pe procesarea tranzacțiilor. Altfel spus: sistemul trebuie să reflecte cât mai fidel realitatea economică, fiind actualizat de îndată ce a survenit o modificare.
Din păcate, sistemul operativ este ineficient atunci când se încearcă folosirea sa pentru a obține informații necesare pentru fundamentarea deciziei economice, informații care să stea la baza analizei economice. Această utilizare a sistemului necesită accesul la toate informațiile într-o manieră "ad-hoc" (ce nu poate fi prevăzută). Dar, sistemul operativ nu este - de regulă - nici unitar și nici omogen (diversele aplicații, uneori rulând pe platforme diferite, utilizează diverse forme și convenții pentru reprezentarea datelor) și, în plus, este optimizat pentru tranzacții și nu pentru interogări masive. În fine, analiza economică mai necesită și foarte multe informații care nu provin din sistemul operațional (de pildă statistici cu caracter general, date demografice rezultate din recensăminte, prognoze meteo și multe altele).
Pentru a rezolva această discrepanță, multe companii dezvoltă și întrețin un sistem informatic paralel, anume destinat pentru a susține analiza economică. Acest sistem, numit "depozit de date" (*data warehouse*) este alimentat cu informații extrase din sistemul operațional precum și din alte surse, informații care sunt "curățate", omogenizate, agregate și stocate într-o maniera care să faciliteze tocmai nevoile de informare ale managementului.
Pentru o descriere mai amplă puteți consulta secțiunea *State Of The Art* din BYTE România nr. 3/1996 (pe web: www.byte.ro/byte96-03/) și articolul "Baze de date multidimensionale" în PC Report, iunie 1997.

(M.S.)

Inserare: o înregistrare nouă se memorează în fișierul de date în mod normal; numărul de ordine al acesteia este ușor de aflat, deoarece toate înregistrările din indexul bitmap au un format fix, și atunci numărul de ordine este chiar numărul de înregistrări din index; în index se completează informațiile detaliate anterior;

Modificare: dacă se modifică unele câmpuri care apar în index, se identifică ușor înregistrarea respectivă pe baza numărului de ordine;

Ștergere: după ce se identifică înregistrarea corespondentă din index, se pune pe 1 bitul corespunzător.

A doua propunere se folosește în conexiune cu tehnica de implementare a B-arborilor. Pentru câmpurile care apar în indexul bitmap se rezervă un câmp suplimentar în B-arbore (de lungime cel puțin egală cu numărul valorilor distincte ale acestora), pe lângă acele informații care identifică o înregistrare din fișierul de date (valoare cheie, adresă înregistrare, lungime).

Valori multiple pentru un atribut

Am văzut că ideea de bază în tehnica de indexare bitmap este de a utiliza un singur bit (în locul unui octet sau mai mulți octeți de date) pentru a indica faptul că o valoare a unui atribut este asociată sau nu unei entități. Cum exploatăm această facilitate în cazul atributelor care pot lua mai multe valori pentru o entitate? De exemplu, dacă dorim să înregistrăm cunoștințele în materie de limbaje de programare (Pascal, C, Fortran, Cobol, Java), asociem atributului *Cunoștințe* 5 biți, fiecare corespunzând unuia din cele cinci limbaje amintite. Un programator poate cunoaște zero (!!!), unul sau mai multe dintre limbajele de mai sus.

Un avantaj al acestei tehnici este acela că spațiul rezervat pentru valorile atributului cheie este folosit efectiv pentru multe înregistrări, spre deosebire de situațiile precedente (atribut monovaloare). Pentru a vedea și alte avantaje ale acestei tehnici să studiem o schiță de program pentru a răspunde la următoarea întrebare:

Care programatori cunosc limbajele Pascal, C și Java?

În primul caz (fără indexare) se consultă doar un singur fișier (cel de date); secvența de interogare arată astfel:

```
cit timp mai există înregistrări
                              executa:
  citeste înregistrare curentă
  daca (Pascal='Da') si
  (C='Da') si (Java='Da')
  atunci
    scrie înregistrare curentă
sfirsit cit timp;
```

În al doilea caz (cu indexare) se consultă două fișiere; indexul bitmap este parcurs în acces secvențial, și fișierul de date este consultat în acces direct numai pentru acele înregistrări care îndeplinesc condiția cerută:

```
cit timp mai există înregistrări
în index executa:
  citeste înregistrare curentă
                          din index
  daca (Limbaj & 11001b) =
                          11001b
  atunci
    citeste înregistrare
          corespondentă din date
    scrie înregistrare citită
sfirsit cit timp;
```

Avantajele tehnicii bitmap

Harta de biți memorată în index conține o reprezentare foarte comprimată a informațiilor conținute în tabela primară. Raportul de comprimare poate fi foarte mare, conducând la economii semnificative ale spațiului ocu-

pat. Această reducere în mărime a intrărilor din tabela index conduce la reducerea numărului de operații de intrare/ieșire în cazul unei interogări complexe. Rezultă că parcurgerea indexului bitmap este mult mai rapidă decât cea a fișierului de date. Să nu uităm că în exploatarea bazelor de date un factor care influențează foarte mult performanțele unei aplicații este folosirea unor tehnici prin care se reduce numărul de operații de intrare/ieșire, acestea fiind foarte costisitoare. În sfârșit, se pot folosi operații logice elementare (și, sau, negație) la nivel de biți pentru a regăsi anumite informații.

Când sunt utili indecșii bitmap?

Atunci când se analizează caracteristicile unei tehnici de indexare, trebuie să avem în vedere mai multe aspecte: performanță, memorie (externă) necesară, ușurința în actualizare. Orice operație de interogare sau actualizare are puncte tari și puncte slabe (am văzut care sunt acestea în cazul B-arborilor). Pentru indecși bitmap, punctele tari constau în procesarea rapidă a unor tipuri de interogări și în cerințe modeste privind spațiul de memorare.

Pentru tehnica de indexare bitmap trebuie să luăm în considerare raportul dintre mărimea înregistrării din fișierul bitmap și înregistrarea din fișierul de date, precum și natura prelucrărilor efectuate în baza de date. Dacă optăm pentru indexare bitmap în situații nepotrivite, se creează o situație ridicolă, pe care înțelepciunea populară o satirizează astfel: "Mai mare daraua decât ocaua".

Dl. Ion Cozac este lector la Universitatea "Petru Maior" din Târgu-Mureș și poate fi contactat prin e-mail la adresa:
cozac@uttgm.ro

Exemplu 1

În tabela cu informații despre clienți (*CLIENTI*), câmpurile *Stare-Civila*, *Zona* și *Sex* pot lua un număr mic de valori, deci se pretează foarte bine la indexare bitmap.

CLIENTI:

Cod Client	Nume	Prenume	Stare-Civilă	Zona	Sex
101	Ionescu	Vasile	necăsătorit	est	B
102	Popescu	Maria	căsătorit	centru	F
103	Dinescu	Aura	divorțat	vest	F
104	Pop	Marin	văduv	sud	B
105	Sava	Irina	căsătorit	nord	F
106	Popescu	Sandu	căsătorit	centru	B

Index pentru CLIENTI:

Cod-Client	Stare-Civila	Zona	Sex
101	1000	00001	10
102	0100	00100	01
103	0010	10000	01
104	0001	00010	10
105	0100	01000	01
106	0100	00100	10

Exemplu 2

2) Tabela PROGRAMATORI cuprinde informații despre limbajele de programare cunoscute de fiecare programator. Câmpurile Pascal, C, Fortran, Cobol și Java pot fi combinate într-un singur câmp în indexul bitmap.

PROGRAMATORI:

Cod-Progr	Nume-Prenume	Pascal	C	Fortran	Cobol	Java
501	Ionescu Vasile	Da	Nu	Da	Da	Nu
509	Popescu Maria	Da	Da	Da	Nu	Nu
620	Dinescu Aura	Nu	Nu	Da	Da	Nu
316	Pop Marin	Da	Da	Nu	Nu	Da
487	Sava Irina	Da	Da	Da	Nu	Da

Index pentru PROGRAMATORI:

Cod-Progr	Limbaj
501	10110
509	11100
620	00110
316	11001
487	11101

OBJECT Pascal și DELPHI

Dragoș Marian

Unul dintre cele mai utilizate limbaje, Pascal, a intrat în era RAD (Rapid Application Development) prin platforma de programare Delphi. Acum, la 4 ani de la prima versiune, Delphi este unul din cele mai folosite limbaje de programare în cadrul firmelor mici.

Cunoașteți, probabil, etapele de bază ale conceperii unui program: analiza problemei, analiza și construirea interfeței, scrierea codului, testarea și depanarea. Delphi a fost gândit astfel încât programatorul care dispune de o analiză serioasă și bine pusă la punct să poată trece rapid peste partea de construire a interfeței, concentrându-se asupra scrierii codului și, în final, a depanării. Cei care ați lucrat pe platforme non-vizuale (Borland Pascal 7 sau Borland C++ 3) știți ce înseamnă asta: interfața construită în trecut într-o săptămână se poate reface astăzi în doar câteva ore cu un instrument vizual (Delphi, Visual C++, Visual Basic etc.). De asemenea, folosirea obiectelor și a claselor este simplificată la maxim.

Delphi, ca toate instrumentele vizuale, este în mod exclusiv bazat pe tehnicile de programare orientată pe obiecte. Fiind un descendent al limbajului Pascal (așchia nu sare departe de trunchi) beneficiază de lizibilitatea foarte bună a codului, prezentând însă două diferențe majore față de Pascalul "tradițional":

În Delphi nu mai există, din punctul de vedere al compilatorului, diferențe între alocarea statică și dinamică a memoriei, adică nu veți mai folosi niciodată simbolul "^".

Câteva instrucțiuni noi, dintre care mai importante sunt două, numite *try..except* și *try..finally*, care se ocupă de excepții. Pe acestea le vom trata mai târziu în cadrul articolului.

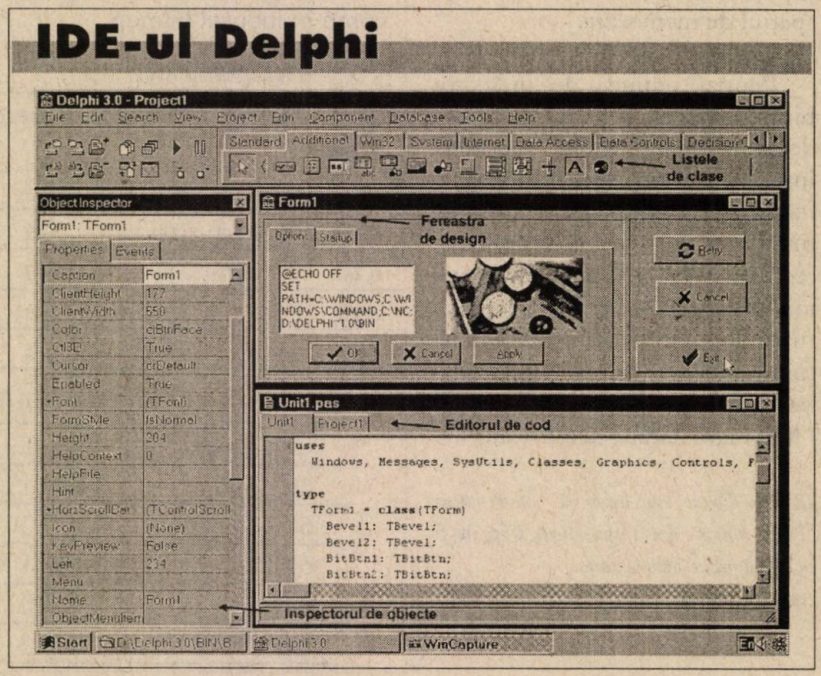
Interfața de operare Delphi

Interfața de operare a Delphi-ului este compusă din patru elemente: tabela de componente (obiecte), inspectorul de obiecte (*Object Inspector*), editorul de cod și editorul de interfață.

Tabela de componente. Este o bară de butoane, fiecare reprezentând o clasă de obiecte. Aceste butoane sunt grupate pe categorii (*Standard*, *Additional*, *Internet* etc.). O categorie foarte importantă este cea a obiectelor standard Windows 95, pe care vă recomand să le folosiți dacă doriți ca aplicația creată să lucreze rapid.

Inspectorul de obiecte (*Object Inspector*) sau mai explicit, editorul de atribute și asocieri, este instrumentul utilizat pentru a modifica atributele unui obiect (textul scris pe un buton, vizibilitatea unei cutii de selectare) și pentru a face legătura între metoda (operația) descrisă în cod de programator și obiectul căruia îi este dedicată metoda.

IDE-ul Delphi



Fiecare obiect al unei clase are un număr de metode și atribute predefinite la care se pot însă adăuga metodele și atributele proprii programatorului, la nivel de clasă.

Editorul de cod este un instrument de lucru cu care sunteți deja obișnuiți de la orele de programare în Pascal sau C. Aspectul diferă puțin (sau mai mult, dacă vreți) față de versiunile în mod text ale platformelor amintite mai sus. Poate vă veți obișnui mai ușor dacă schimbați culorile în combinația clasică Borland de albastru, alb și galben (în meniul *Tools/ Environment Options* la secțiunea *Colors/Color SpeedSetting/ Classic*, sau - mai simplu - cu un clic-dreapta pe editor în meniul *Properties*). Obișnuitele Breakpoint-uri (puncte în care execuția programului este oprită în cazul depanării) se fac cu un simplu clic pe editor în stânga liniei la care dorim plasarea acestuia.

Editorul de interfață formează, împreună cu tabela de componente și inspectorul de obiecte, "baza" proiectării vizuale. Este o fereastră (*Form*) pe care se așează componentele vizuale și nonvizuale cu care va lucra programul. Fiecărei ferestre îi corespunde un *unit* care conține, pe lângă funcțiile și procedurile proprii programatorului, descrierile metodelor implementate de programator și folosite de fereastră precum și obiectele de pe aceasta. Pentru plasarea acestora (de exemplu plasarea unui buton) se selectează din tabela de componente clasa din care face parte obiectul pe care dorim să-l folosim și se specifică (printr-un clic) fereastra în care trebuie plasat. În această fereastră va apare un obiect - în cazul nostru un buton - al cărui atribute și asocieri cu metodele folosite de acesta le putem modifica din inspectorul de obiecte. Atenție! Este foarte important de reținut că din paleta de componente nu se selectează un anumit obiect, ci o *clasă de obiecte* care va fi *reprezentată în fereastră de un obiect!* Două atribute specifice obiectelor vizuale, dimensiunea și poziția în fereastră, pot fi modificate direct din fereastră, fără a folosi inspectorul de obiecte, în mod similar cu aplicarea aceleiași operații pe o fereastră a unui program Windows. Este foarte important faptul că o fereastră este ea însăși o componentă (obiectul *Form1* aparține clasei *TForm*). O nouă fereastră se creează cu ajutorul unui buton special, în stânga paletei de componente.

Listing 1 - Secțiunea Private

```
unit HWForm;
uses SysUtils, WinTypes, WinProcs, Messages, Classes,
Graphics, Controls, Forms, Dialogs,

type
THelloWorldForm = class(TForm){ declară clasă nouă de tip fereastră }
procedure FormCreate(Sender:TObject);
private                                  { partea invizibilă a codului }
HelloWorldString:string                  { declară un câmp ascuns }
end;

var
HelloWorldForm:THelloWorldForm;
implementation
procedure THelloWorldForm.FormCreate(Sender:TObject);
begin
HelloWorldString:='Hello World'
end
end.

                                  { Testăm clasa THelloWorldForm }
unit TestHelloWorldForm;                  { Fereastra principală }
interface
uses SysUtils, WinTypes, WinProcs, Messages, Classes,
Graphics, Controls, Forms, Dialogs,
HWForm;                          { declarăm utilizarea unitului clasei }
type
TTestForm = class(TForm)
procedure FormCreate(Sender:TObject)
end;
var
TestForm:TTestForm;
implementation
procedure TTestForm.FormCreate(Sender:TObject);
begin
HelloWorldForm.HelloWorldString :='Invizible Hello World'
{ Compilatorul se oprește cu mesajul "Field identifier expected" }
end
end.
```

Structura unui proiect Delphi

În Delphi s-a trecut la o structurare a codului aplicației oarecum asemănătoare cu cea a proiectelor din C++. Astfel, partea cea mai importantă a unui program o constituie fișierul proiect (*nume_proiect.prj*) care conține toate asocierile fereastră-unit aparținând programului. De asemenea, conține toate rutinele de inițializare și creare ale acestora. Generarea acestui fișier se face în mod automat, rezervând programatorului posibilitatea de a modifica ordinea de creare a ferestrelor, distrugerea acestora făcându-se tot în mod automat. De asemenea, fișierul-proiect conține datele referitoare la fereastra principală (*Main Form*), care este prima fereastră afișată pe ecran din punctul de vedere al

Listing 2 - Secțiunea Protected

```
unit HWForm;
uses SysUtils, WinTypes, WinProcs, Messages, Classes,
Graphics, Controls, Forms, Dialogs,

type
THelloWorldFormSuperClass = class(TForm)
                              { declară clasă nouă de tip fereastră }
procedure FormCreate(Sender:TObject);
protected                                  { partea invizibilă a codului }
HelloWorldString:string                    { declară un câmp ascuns }
end;

var
HelloWorldForm:THelloWorldFormSuperClass;

implementation
procedure THelloWorldForm.FormCreate(Sehder:TObject);
begin
HelloWorldString:='Hello World'
end;
end.

                    { Construim clasa THelloWorldFormSubClass }

unit TestHelloWorldFormSuperClass;          { Fereastra principală }

interface
uses SysUtils, WinTypes, WinProcs, Messages, Classes,
Graphics, Controls, Forms, Dialogs,
HWForm;                          { declarăm utilizarea unitului clasei }

type
THelloWorldFormSubClass = class(TForm)
                              { declară clasă nouă de tip fereastră }
procedure FormCreate(Sender:TObject);
protected                          { partea protejată a codului nostru }
MyHelloWorldString:string                  { declară un câmp ascuns }
end;

var
TestForm:THelloWorldFormSubClass;

implementation
uses HWForm;

procedure TTestForm.FormCreate(Sender:TObject);
begin
HelloWorldForm.MyHelloWorldString :=
HelloWorldForm.HelloWorldString + 'I love you'
          { HelloWorld și MyHelloWorldString sunt vizibile pentru }
                { programatorul clasei THelloWorldFormSubClass }
end
end.
```

utilizatorului. Aceasta este prima fereastră creată.

În fișierul proiect se pot include comenzi de afișare a unor ferestre (*Splashscreen*) care să apară pe ecran în timpul creării restului ferestrelor.

Manipularea unei ferestre în timpul execuției programului se face din *unit*-ul asociat ferestrei respective. Astfel, fiecare obiect are o suită de proprietăți care pot fi apelate și modificate din program sau din inspectorul de obiecte. De asemenea, un obiect are o suită de evenimente (*events*), listate pentru fiecare obiect în inspectorul de obiecte. Un ***eveniment*** este o asociere obiect-procedură, asociere care poate fi declanșată de interacțiunea cu utilizatorul, de modificarea unui atribut, de apariția unei erori etc. În Delphi, erorile sunt considerate obiecte ale unor clase speciale, numite ***excepții*** (*exceptions*) care conțin asocieri între erori și un mesaj standard de eroare. Este permisă modificarea acestor excepții prin instrucțiunile *try..except* și *try..finally*. Unele componente au evenimente prin care se pot modifica aceste excepții fără folosirea acestor instrucțiuni. De exemplu, un obiect aparținând clasei *TTable* moștenește evenimentul *OnDeleteError* care, dacă nu este asociat unei proceduri implementate de programator, generează un mesaj de eroare de tipul *"Project MyApp.exe has raised exception class EDeleteError with message 'Record Delete Error'"* - extrem de criptic pentru utilizatorul final.

Crearea unei asocieri eveniment-metodă se face cu ajutorul inspectorului de obiecte. O asociere se poate face simplu prin clic dublu pe evenimentul căruia dorim să-i asociem o procedură (de exemplu, butoanele au evenimentul *OnClick* - în timpul execuției programului: un clic pe butonul căruia i-a fost asociată o procedură la acest eveniment va determina executarea codului scris în acea procedură). Procedura va face parte din *unit*-ul asociat ferestrei în care este obiectul respectiv și va fi declarată automat, programatorul urmând doar să implementeze codul.

Utilizarea atributelor obiectelor

Există două concepte de bază pentru înțelegerea relațiilor dintre clase: moștenirea și generalizarea. Acestea sunt abstracții puternice folosite pentru transmiterea similarităților dintre clase (atribute și metode comune), păstrând totuși diferențele dintre ele.

Generalizarea se definește ca fiind relația dintre o clasă de bază și una sau mai multe clase derivate (versiuni ale clasei de bază). În teoria programării orientate pe obiecte, clasa de bază se numește *superclasă* iar clasele derivate se numesc *subclase*. Toate atributele și metodele comune sunt grupate în clasa de bază și sunt "moștenite" de clasele derivate.

Clasele derivate sunt versiuni specializate ale clasei de bază. Există o clasă, să o numim strămoșul general, care nu este derivată din nici o altă clasă. Este, dacă vreți, clasa cea mai de bază. Din această clasă încep ramificațiile claselor derivate, care se ramifică la rândul lor în clase tot mai specializate, formându-se o structură arborescentă numită ierarhie.

Un obiect posedă toate atributele și metodele clasei din care face parte. Toate clasele Delphi derivă din clasa de bază *TObject* (în Delphi, prezența literei *T* sau *E* în fața unei denumiri semnifică faptul că este vorba de o clasă de obiecte, respectiv de erori). Această clasă de obiecte conține doar două atribute, referitoare la poziția în fereastră a obiectului, considerate necesare fiecărei clase în parte: distanța față de marginea de sus (Top) și distanța față de marginea din stânga (Left) a ferestrei.

Să presupunem că folosim o instanță a clasei *TListBox* (listă de selecție) numită *MyListBox* și dorim ca în timpul programului să adăugăm un element (un string) în această listă. Sintaxa folosită este următoarea:

```
MyListBox.Items.
  Add('Hello World');
```

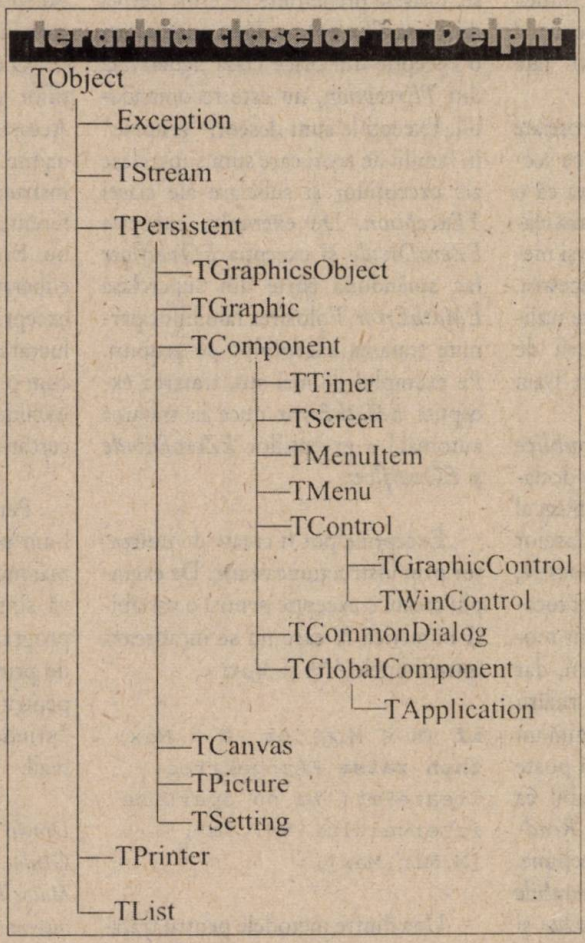
După cum se observă, atributele se despart de numele instanței prin punct, iar atributele pot avea și ele sub-atribute, care de cele mai multe ori sunt metode. În exemplul dat, atributul *Items* (elemente) are metoda de adăugare în lista de elemente numită *Add* care necesită un parametru de tip string. Acum să presupunem că dorim ștergerea celui de-al treilea element din listă. Deoarece numerotarea începe de la zero, folosim instrucțiunea:

```
MyListBox.Items.Delete(2);
```

Din exemplu se poate vedea că metoda *Delete* necesită un parametru de tip întreg mai mare sau egal cu zero.

Pentru folosirea metodelor și atributelor din program, fără ajutorul inspectorului de obiecte, începând cu Delphi versiunea 3 există un instrument foarte util. Acesta se cheamă *Code Insight* și vă va scuti de căutări lungi prin fișierele de help.

Ierarhia claselor în Delphi



Cum funcționează? Mai întâi, asigurați-vă că ați selectat toate opțiunile din caseta de dialog *Environment Options/Code Insight*. Dacă acestea sunt selectate, la tastarea în editorul de cod a numelui unei instanțe sau atribut urmată de punct va apărea, după 0.5..1.5 secunde (în funcție de selectare), o listă cu toate atributele și metodele folosibile. De asemenea, la folosirea unei metode (incluzând și funcțiile și procedurile care nu țin de obiecte) care are nevoie de parametri, tastarea numelui acesteia urmată de "(" vă va arăta lista de parametri.

Crearea claselor de obiecte

Definiția unei clase de obiecte se poate face în două moduri: local și general. "Local" înseamnă că vom folosi clasa creată doar în cadrul programului în care o vom crea. "General" înseamnă că vom putea dispune de clasa creată și în alte programe, prin înregistrarea ei în paleta de componente.

Definiția unei clase este aceeași pentru amândouă modurile. Diferența constă în faptul că o clasă locală este creată manual într-unul din *unit*-urile proprii aplicației în care va fi folosită, pe când clasele generale sunt vizibile în orice program. Nu există o delimitare fixă între local și general. O clasă locală se poate transforma într-una generală prin adăugarea unei proceduri de înregistrare.

O clasă nu poate fi definită decât în secțiunea *type* a unitului. Declararea trebuie să țină cont de clasa de bază căreia îi va aparține (de exemplu, în listingul 1 am creat o clasă de ferestre, bazată pe clasa de ferestre predefinită de sistem).

O clasă are patru secțiuni de declarații pentru atributele și metodele implementate:

Secțiunea de declarații ascunse (*private*). Declararea unui atribut sau metodă în această secțiune îl face vizibil doar în cadrul *unit*-ului în

Listing 3

```
try
 ...
except
 on EZeroDivide do MesajDivZero;
 on EOverflow do MesajOverflow;
else
 AfiseazaMesaj('Eroare
netratata')
end;
```

Listing 4

```
reset(fisier);
try
read(fisier, date);
finally
CloseFile(fisier)
end;
```

care este declarat. În afara acestuia atributul sau metodă este invizibil. În listingul 1 se poate vedea cum compilatorul generează o eroare deoarece nu găsește stringul *HelloWorldString* care a fost declarat în secțiunea *private*. Este o secțiune utilă pentru a ascunde detaliile implementării față de utilizatorii obiectului creat.

Secțiunea de declarații protejate (*protected*). Este asemănătoare cu secțiunea anterioară, cu deosebirea că o clasă construită în alt *unit* pe baza clasei în care am introdus atribute și metode protejate are acces la acestea. Astfel se face diferențierea între utilizatorii unui obiect și creatorii de obiecte pe baza obiectului care l-am creat noi (vezi listingul 2).

Secțiunea de declarații publice (*public*). Atributele și metodele declarate aici sunt vizibile oricărui obiect al clasei respective, respectiv claselor create ca descendenți ai clasei noastre, indiferent de *unit*-ul în care sunt localizate, cu specificarea că nu sunt modificabile în timpul programării, dar au un anumit rol în timpul rulării programului (de exemplu, definim un mesaj de eroare și nu dorim să poate fi modificat). Este recomandabil ca datele ce sunt în mod necesar *Read-Only* să fie declarate în această secțiune.

Secțiunea de declarații alterabile (*published*). Declararea atributelor și metodelor în această secțiune păstrează vizibilitatea ridicată a acestora. De asemenea, atributele declarate aici pot fi modificate prin inspectorul de obiecte (devenind vizibile prin intermediul acestuia) sau în timpul execuției programului.

Deci definiția unei clase ar trebui să arate în modul următor:

```
type TmyClass =
     class(TAncestor Class)
Private
{ Declarații ascunse }
Protected
{ Declarații protejate }
Public
{ Declarații publice }
Published
{ Declarații alterabile }
end;
```

Tratarea excepțiilor

După cum am spus mai devreme, o excepție este o clasă de erori. Aceste excepții se definesc ca oricare alte clase. Clasele predefinite de erori derivă din clasa *TException*. Deși putem crea o excepție din orice clasă nederivată din *TException*, nu este recomandabil. Excepțiile sunt deseori "grupate" în familii de erori care sunt superclase ale excepțiilor și subclase ale clasei *TException*. De exemplu, excepția *EZeroDivide* și excepția *EOverflow* fac amândouă parte din superclasa *EMathError*. Folosirea familiilor permite tratarea excepțiilor pe grupuri. Pe exemplul de mai sus, tratarea excepției *EMathError* duce la tratarea automată a excepțiilor *EZeroDivide* și *EOverflow*.

Excepțiile pot fi create de utilizator prin instrucțiunea *raise*. De exemplu creăm o excepție pentru o variabilă numerică *N* care nu se încadrează între limitele *Min* și *Max*:

```
if (N < Min) or (N > Max)
then raise ERangeError.
CreateFmt('%d nu apartine
intervalului %d..%d',
[N,Min,Max]);
```

Una dintre metodele pentru tratarea excepțiilor este instrucțiunea *try..except*. Sintaxa acestei instrucțiuni constă din două blocuri:

```
try
{ Operații }
except
{ Tratarea excepțiilor posibile }
{ la operațiile de mai sus }
[else]
{ Cazul în care nu avem }
{ o tratare a excepției }
end;
```

În momentul întrării în instrucțiune se execută operațiile din secțiunea *try*. Dacă nu se întâlnește nici o eroare, se sare peste secțiunea de tratare a excepțiilor. Dacă se întâlnește o excepție, se caută în secțiunea *except* un *handler* (manipulator) pentru aceasta. Dacă nu este găsit și nu există secțiunea opțională *else*, se va activa excepția standard a Delphi-ului (vezi listingul 3).

Codul din secțiunea de tratare a excepțiilor poate crea excepții proprii prin instrucțiunea *raise*.

O altă metodă de tratare a excepțiilor este instrucțiunea *try..finally*. Aceasta este asemănătoare cu cea dinainte, cu particularitatea că blocul de instrucțiuni *finally* este executat indiferent dacă a avut loc a excepție sau nu. Este foarte utilă pentru a asigura eliberarea unor resurse indiferent de excepțiile întâlnite de acestea în prelucrare. În listingul 4, puteți vedea cum o instrucțiune *try..finally* asigură închiderea unui fișier după ce s-a încercat citirea unui element.

Pentru a acoperi subiectul pe care l-am propus în titlu, ar fi nevoie de mai mult de un articol. Am încercat să vă sintetizez principiile de bază ale programării în Delphi, și să vă scutesc de primele ore de "săpat" prin Help pentru a înțelege IDE-ul. Dacă vă "prindeți urechile", trimiteți-mi un e-mail.

Dragoș Marian este colaborator al revistei GInfo, student la Universitatea Petru Maior Tg-Mureș, și poate fi contactat prin intermediul redacției: office@ginfo.ro

Intrări/ieșiri și COMPILARE în Java

Horia Georgescu

Intrările/ieșirile sunt simple!? Și totuși multe neplăceri ne provoacă. Ce facilități oferă Java de a acționa asupra unui fișier de intrare în scopul detectării entităților sale?
Pe de altă parte a sosit timpul să tratăm câteva amănunte și despre compilarea programelor Java, respectiv despre lucrul cu pachete în Java

Facilități de intrare/ieșire

Începem prin prezentarea clasei `input`, anunțată în numerele anterioare, dar din păcate omisă:

Listing 1 - Clasa Input

```
import java.io.*;
public class input {
  public static void readln() {
    try { System.in.read(); }
    catch(IOException e) {
      System.out.println("Input Error");
    }
  }

  public static double
              read(StreamTokenizer f) {
    try {
      f.nextToken();
      if (f.ttype == -2)
return(f.nval);
      else return(Double.NaN);
    }
    catch(IOException e)
    { System.out.println("Input Error");
    return(0.0);\}
  }
}
```

Clasa `input` pune la dispoziția utilizatorului următoarele metode:

- metoda `readln()`, care citește un caracter din șirul de caractere de la intrarea standard;
- metoda `read`, care întoarce o valoare numerică de tip **`double`** (dacă este detectată o astfel de valoare) sau constanta `isNaN` (în caz contrar).

Explicațiile necesare sunt furnizate în continuare.

Clasa StreamTokenizer

Clasa `StreamTokenizer` din pachetul `java.io` permite, prin intermediul câmpurilor și metodelor sale, să acționăm asupra unui fișier de intrare în scopul detectării *entităților* sale; prin urmare suntem în prezența unui *analizor lexical.*

Entitățile care pot fi identificate în mod standard sunt:

- identificatorii;
- șirurile de caractere;
- sfârșitul de fișier (`EOF`).

Opțional, mai pot fi identificate și următoarele entități:

- sfârșitul de linie (`EOL`);
- comentariile "de sfârșit de linie" (care încep cu `"//"` și continuă până la sfârșitul liniei;
- comentariile "cu asterisc" (care încep cu `"/*"` și se termină cu prima succesiune `"*/"`).

Observație: eventuala identificare a comentariilor este însoțită de ignorarea lor (deci nu de furnizarea conținutului lor).

Fiecare octet citit din fișierul de intrare este privit ca un caracter din intervalul delimitat de `'\u0000'` și de `'\u00FF'` și este considerat ca având nici unul, unul sau mai multe dintre următoarele atribute: spațiu alb al analizorului, caracter alfabetic, caracter numeric, identificator de șiruri (apostrof sau ghilimele) și caracter comentariu.

Clasa `StreamTokenizer` conține următoarele câmpuri, declarate cu modificatorii **`public static`**

```
final int:

TT_EOF = -1;     TT_EOL = '\n';
TT_NUMBER = -2; TT_WORD = -3
```

corespunzătoare în ordine entităților: *sfârșit de fișier, sfârșit de linie, valoare numerică* și *identificator*.

Constructorul clasei `StreamTokenizer` are un parametru de tipul `InputStream`; nu vom intra în detalii: deocamdată acest parametru va fi `System.in`, corespunzător dispozitivului standard de intrare (tastatură). Constructorul realizează următoarele inițializări (definiri prin lipsă):

- caracterele de la `'a'` la `'z'`, de la `'A'` la `'Z'` și de la `0xA0` la `0xff` sunt considerate a fi caractere alfabetice;
- caracterele de la `0x00` la `0x20` sunt considerate a fi spațiile albe ale analizorului; să observăm că sunt incluse aici spațiile albe uzuale (*spațiu, tab, newline, formfeed*);
- caracterul "`/`" este caracter comentariu;
- apostroful și ghilimelele sunt considerate identificatori de șiruri;
- `EOL` nu este recunoscut (este ignorat);
- comentariile nu sunt recunoscute.

Tipul entității identificate va fi regăsit în câmpul `ttype`, declarat cu **`public int`**. Valorile pe care le poate lua `ttype` sunt cele constante de mai sus, plus o valoare nenegativă; ne mărginim la a menționa că dacă a fost identificat un șir de caractere, această valoare corespunde lui `'\"'`.

Câmpul `sval`, decalarat cu **`public String`**, memorează un identificator sau un șir de caractere, în situația în care a fost detectată o astfel de entitate; în caz contrar, valoarea lui `sval` este `null`.

Câmpul `nval`, declarat cu **`public`** `double`, are semnificație dacă a fost recunoscută o valoare numerică de tip **`double`**, caz în care memorează acea valoare. Precizăm că este vorba numai de numere zecimale cu sau fără punct; analizorul nu recunoaște `31.4e-1` ca număr în virgulă mobilă și nici `0xeee` ca număr hexazecimal.

Dacă se dorește identificarea sfârșiturilor de linie, a comentariilor de sfârșit de linie și a celor cu asterisc, trebuie apelate în prealabil următoarele metode cu specificarea ca parametru a lui `true`:

```
EolIsSignificant
slashStarComments
slashSlashComments
```

Metodele `ordinaryChar`, `commentChar` și `quoteChar` au un parametru de tipul **`int`**. Apelarea lor pentru un caracter `ch` face ca, respectiv, `ch` să devină un caracter obișnuit (fără vreun atribut), un caracter comentariu sau un un caracter pentru identificarea șirurilor de caractere.

De asemenea mai menționăm metodele `ordinaryChars`, `whitespaceChars` și `wordChars` ce au două argumente (fie ele `p` și `u`) de tipul **`int`**. Prin apelarea lor, caracterele din secvența `p..u` capătă respectiv atributele caracter obișnuit (fără atribute), spațiu alb al analizorului, atributul alfabetic.

Dar metoda centrală a clasei `StreamTokenizer` este:

```
public int nextToken() throws IOException;
```

a cărei invocare are drept efect detectarea următoarei entități din șirul de caractere de la intrare. Acțiunile întreprinse sunt descrise în continuare.

Mai întâi sunt ignorate spațiile albe ale analizorului, cu următoarea excepție: dacă este întâlnit sfârșitul de linie și acesta este recunoscut de analizor, atunci entitatea detectată va fi `TT_EOL`.

Dacă este întâlnit un caracter numeric, se încearcă detectarea unei valori numerice. Dacă respectivul caracter este `'-'` dar următorul este nenumeric, atunci entitatea detectată va fi acest caracter. În caz contrar, este detectată o valoare numerică, avansând în șirul de intrare până la întâlnirea unuia dintre următoarele caractere: al doilea `'-'`, al doilea `'.'`, primul caracter nenumeric sau sfârșitul de fișier; drept urmare entitatea detectată este `TT_NUMBER`, iar valoarea ei se regăsește în câmpul `nval`.

Dacă este întâlnit un caracter alfabetic, va fi detectat identificatorul delimitat la dreapta de primul dintre următoarele tipuri de caractere: nealfabetic, nenumeric sau sfârșit de fișier.

Dacă este întâlnit un caracter cu atributul comentariu, atunci toate caracterele până la sfârșitul de linie sau de fișier sunt ignorate, cu următoarea excepție: dacă este întâlnit sfârșitul de linie și acesta este recunoscut de analizor, atunci entitatea detectată va fi `TT_EOL`.

Dacă este întâlnit un caracter identificator de șir de caractere, entitatea detectată va fi `TT_WORD`, iar valoarea ei poate fi regăsită în `sval`. Valoarea constă din toate caracterele până la primul dintre următoarele: același caracter identificator de șir, sfârșit de linie sau sfârșit de fișier.

Dacă este întâlnită secvența `//` și analizorul recunoaște comentariile de sfârșit de linie, atunci sunt ignorate toate caracterele până la primul sfârșit de linie sau fișier.

Dacă este întâlnită secvența `/*` și analizorul recunoaște comentariile "cu asterisc", atunci sunt ignorate toate caracterele până la prima apariție a secvenței `*/` sau până la primul sfârșit de fișier.

Dacă nu suntem în nici una dintre cazurile enumerate, atunci suntem în prezența unui caracter obișnuit, entitatea întoarsă fiind `TT_WORD`.

Exemplificăm descrierea de mai sus prin următorul program:

Listing 2 - Clasa Token

```
import java.io.*;class token {
  public static void main(String[] s)
                    throws IOException {
    StreamTokenizer st = new
            StreamTokenizer(System.in);
    st.ordinaryChar('/');
    st.slashStarComments(true);
    st.slashSlashComments(true);
    st.eolIsSignificant(true);
    for (int entitate = st.nextToken();
      entitate !=StreamTokenizer.TT_EOF;
      entitate = st.nextToken()) {
   if (st.ttype == -2)
    System.out.println("Numar: "+st.nval);
   else if (st.ttype == -3)
System.out.println("Identif.: "+st.sval);
   else if (st.ttype == '\"')
   System.out.println("Sir:  "+st.sval);
   else if (st.ttype == '\n')
   System.out.println("EOL");
    }
  }
}
```

Ne rezumăm la a menționa că pentru recunoașterea celor două tipuri de comentarii a fost necesar să privăm caracterul '/' de atributul conferit de constructorul clasei StreamTokenizer.

Fișiere cu nume

Vom prezenta în continuare o modalitate de exploatare a fișierelor cu nume formate din succesiuni (fluxuri) de caractere, analoage fișierelor text din Pascal.

Pentru a deschide un astfel de fișier și a folosi facilitățile oferite de clasa StreamToikenizer vom folosi instrucțiunile:

```
FileInputStream ff = new
            FileInputStream(nume_fisier);
StreamTokenizer f= new StreamTokenizer(ff);
```

Ieșirea într-un "fișier text" se realizează prin:

```
FileOutputStream gg = new
            FileOutputStream(nume_fișier);
PrintStream g = new PrintStream(gg);
```

ceea ce permite ca în continuare să folosim metodele print și println la fel ca pentru System.out.

În ambele cazuri poate apare excepția (eroarea) FileNotFoundException.

Exemplul 1.

Programul următor citește din fișierul cu numele ion valori numerice până la prima entitate nenumerică și le scrie pe cele cu valoare pozitivă în fișierul vasile, creat cu acest prilej. Aceste două fișiere apar în același director cu in_out.class.

Listing 3 - Fișiere cu nume

```
import java.io.*; import input;
import java.lang.*;
class in_out {
  public static void main(String arg[])
  throws IOException,FileNotFoundException {
    double val;
    FileInputStream ff = new
                  FileInputStream("ion");
    StreamTokenizer f = new
                  StreamTokenizer(ff);
    FileOutputStream gg = new
            FileOutputStream("vasile");
    PrintStream g = new PrintStream(gg);
    val = input.read(f);
    while( ! Double.isNaN(val)) {
      if (val>0) g.print(val+" ");
      val = input.read(f);
    } g.close();
  }
}
```

Exemplul 2.

Programul următor citește din fișierul cu numele ionica toate caracterele până la sfârșit de fișier și scrie în fișierul vasilica, creat cu acest prilej, numai literele mici întâlnite. De această dată nu se mai face apel la clasa StreamTokenizer, ci este folosită metoda read a clasei InputStream, al cărei efect este următorul: este citit un byte și este întoarsă valoarea sa, cuprinsă între 0 și 255; dacă s-a detectat sfârșitul de fișier, este întoarsă valoarea -1.

Listing 4 - Fișiere cu nume

```
import java.io.*;
class in_out_c {
  public static void main(String arg[])
  throws IOException,FileNotFoundException {
    int i; char ch;
    FileInputStream ff = new
              FileInputStream("ionica");
    FileOutputStream gg = new
          FileOutputStream("vasilica");
    PrintStream g = new PrintStream(gg);
    i = ff.read(); ch =(char) i;
    while( i>=0) {
      if (ch>='a'&& ch<='z') g.print(ch);
      i = ff.read(); ch = (char) i;
    } g.close();
  }
}
```

Compilarea programelor Java

Compilatorul Java acționează asupra fișierelor sursă Java (numite și *unități de compilare*), identificate

prin extensia `.java`. Un astfel de fișier poate conține una sau mai multe clase sursă. Dintre acestea cel mult una poate fi declarată cu modificatorul **public**, celelalte fiind considerate ca având un modificator implicit fără nume, la care ne vom referi prin "pachet"; rolul acestor modificatori va fi prezentat mai jos. Dacă într-un fișier sursă apare o clasă declarată **public**, atunci numele fișierului trebuie să coincidă cu numele clasei.

Programele executabile `javac.exe` și `java.exe` se află în subdirectorul `bin` al directorului în care a fost instalat Java (vom presupune în continuare că numele acestui director este `jdk1.1.5`).

La compilarea unui fișier sursă sunt create atâtea fișiere cu extensia `.class` câte clase sursă există în fișier; numele lor coincide cu numele claselor sursă. Aceste fișiere vor fi numite, pentru prescurtare, *clase obiect*. O clasă obiect reprezintă codul binar atașat unei clase sursă și va avea același nume (dar bineînțeles altă extensie).

Clasele obiect sunt plasate în mod standard în directorul în care apare fișierul sursă.

Pentru primele programe scrise în Java, modul tipic de lucru este de a plasa toate fișierele într-un același director; vom presupune că numele acestuia este `progjava`. Să remarcăm că în acest mod nu este necesar să compilăm totdeauna separat fiecare clasă: de exemplu, includerea directivei: **import** `input;` într-un fișier sursă face ca, la compilarea sa, clasa `input` să fie și ea compilată (dacă nu există deja clasa obiect `input.class`).

Acest mod de lucru devine însă nemulțumitor imediat ce numărul programelor crește. La aceasta se adaugă faptul că în mod firesc putem dori să grupăm mai multe clase într-o structură aparte, care poartă în Java numele de *pachet*.

Lucrul cu pachete în Java

Un *pachet* are forma unui director ce conține codurile binare (clasele obiect) asociate mai multor clase sursă; vom presupune că numele directorului este `pachet`. Modul de lucru cu pachete, descris în continuare, face să apară clar faptul că un pachet nu se reduce la structura sa de director.

Conținutul tipic al unui pachet este următorul:

- clasa obiect corespunzătoare clasei sursă declarate cu **public**;
- clasele obiect corespunzătoare claselor sursă "declarate" cu modificatorul implicit (adică fără vreunul dintre modificatorii de acces **public**, **private**, **protected**).

Ideea care stă la baza acestei structuri este următoarea: din afara pachetului putem avea acces doar la clasa declarată **public**, iar din interiorul acesteia avem acces la celelalte clase ale pachetului. Vom presupune că numele clasei declarate **public** este `ClasaPub`. Structura de mai sus a unui pachet nu este însă obligatorie: de exemplu un pachet poate conține rezultatul compilării mai multor fișiere sursă Java, în particular poate conține mai multe clase cu modificatorul **public**.

Prima linie a fișierului sursă ce conține clasele menționate trebuie să conțină declararea: **package** `pachet;` care dă informații compilatorului asupra numelui pachetului.

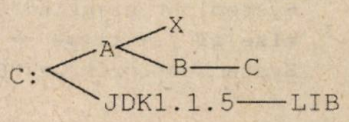
Dacă un fișier sursă face referire la clasa `ClasaPub`, atunci ca prima linie a acestui fișier trebuie plasată directiva **import** :

import `pachet.ClasaPub;`

Dacă dorim acces la mai multe clase ale pachetului, vom folosi directiva sub următoarea formă:

import `pachet.*;`

Vom considera următoarea structură de directoare, unde în directorul `C` apare fișierul `input.java`.



Presupunem că ne aflăm în directorul `C:\A\X`, unde trebuie compilat fișierul sursă `Comb.java`, fișier ce intenționează să utilizeze clasa `input`. Pentru aceasta vom proceda astfel:

- în fișierul `input.java` plasăm ca primă linie directiva: **package** `B.C;`
- în fișierul `Comb.java` introducem ca primă linie directiva: **import** `B.C.input;`
- compilăm fișierul `Comb.java` cu:

```
javac -classpath
c:\jdk1.1.5\lib\classes.zip;.. Comb.java
```

- comandăm executarea clasei obiect `Comb.class` cu:

```
java -classpath
    c:\jdk1.1.5\lib\classes.zip;.;.. Comb
```

Este folosită opțiunea `-classpath` urmată de o listă de argumente separate prin punct și virgulă.

Explicațiile sunt următoarele:

- nelucrând într-un subdirector al lui `jdk1.1.5`, trebuie asigurat accesul (atât pentru compilator, cât și pentru interpretor) la "clasele esențiale" furnizate de Java, a căror locație standard este :
  `c:\jdk1.1.5\lib\classes.zip`
- informațiile din **package** și **import** urmăresc ca, mergând în sus în arborele determinat de structura de subdirectoare, să se ajungă la un subdirector comun (în exemplul nostru este vorba de `C:\A`);
- la lansarea interpretorului, trebuie precizat și directorul curent, pentru a avea acces la clasa obiect `Comb.class`.

Prof. dr. Horia Georgescu este profesor la Facultatea de Matematică din București, e-mail: hg@colegiu.cs.unibuc.ro

Pagini WEB cu HTML 3.2

Mihai Scorțaru

Vreți să transmiteți cartea de vizită, fotografia, hobby-urile și multe altele lumii întregi? Soluția este simplă! Nu vă trebuie decât un calculator conectat la Internet, câteva cunoștințe privind HTML... adică trebuie să citiți articolul care urmează.

Câteva date istorice

Cu mai mulți ani în urmă, mai mulți oameni de știință s-au gândit că ar fi bine ca cele mai importante rețele de calculatoare să fie conectate între ele. Rezultatul a fost apariția "mamei tuturor rețelelor", numită acum *Internet*. Până în anul 1990 accesarea informațiilor de pe *Internet* era foarte dificilă și rețeaua nu era folosită decât de un grup restrâns de oameni de știință. Fizicianul Tim Berners-Lee a devenit celebru datorită faptului că a "inventat" link-urile (legături, referințe) hypertext. Deși această idee nu era nouă, ea a dus la dezvoltarea unui limbaj simplu care s-a impus până la urmă ca standard. Tim Berners-Lee a numit acest limbaj Hypertext Markup Language, cunoscut sub denumirea prescurtată de HTML. Noțiunea de hypertext înseamnă text păstrat în format electronic cu link-uri între pagini.

În anul 1993 în jur de 100 de calculatoare din lumea întreagă erau echipate pentru a găzdui pagini HTML. Aceste pagini interconectate au fost "poreclite" World Wide Web. Nu mult timp după aceea au început să fie scrise primele browsere Web cu ajutorul cărora puteau fi vizualizate pagini Web care conțineau text și imagini. La ora actuală paginile Web pot fi vizualizate și cu ajutorul televizorului, dar încă nu pot fi concepute fără ajutorul unui calculator.

În funcție de versiunea HTML folosită, paginile Web sunt grupate în trei "generații".

- Paginile din prima generație, care foloseau HTML 1.0, conțin în cea mai mare parte text și o imagine sau două atașate textului.
- A doua generație de pagini Web, proiectate cu ajutorul versiunii HTML 2.0, permite adăugarea unui fundal, aranjarea textelor în tabele și posibilitatea de a comanda un produs prin intermediul Internetului (comandă on-line).
- A treia generație de pagini Web folosește versiunea HTML 3.2 care a devenit standardul pentru proiectarea paginilor Web. Aceste pagini pot conține culori definite de utilizator, secvențe animate, sunete etc.

Unelte necesare pentru crearea și vizualizarea paginilor Web

Pentru a crea și vizualiza pagini Web este nevoie de un calculator conectat la Internet, un editor de texte și un browser Internet. Pentru a putea lucra în condiții acceptabile, vă recomandăm să folosiți un calculator Pentium MMX la 166 MHz sau unul cu o configurație superioară. Legătura Internet ar trebui să fie printr-un modem care să aibă o viteză de 56 Kbps (Kilobiți pe secundă) pentru a asigura o încărcare mai rapidă a paginilor de pe Internet. 90% dintre cei care navighează pe Internet folosesc unul dintre browserele Netscape Navigator sau Internet Explorer așa că, este recomandabil ca ambele să fie prezente pe calculatorul dumneavoastră. Ele pot fi preluate gratuit de pe Internet de la adresele: http://home.netscape.com, respectiv http://www.microsoft.com. Codul HTML poate fi editat cu ajutorul oricărui editor de texte care permite salvarea fișierului în format TEXT. Pentru a vizualiza o pagină Web trebuie să folosiți comanda *Open* din meniul *File* al browserului Internet pe care îl folosiți. În fereastra de dialog care va apărea trebuie ales fișierul care conține codul HTML corespunzător.

Elemente de bază pentru un fișier HTML

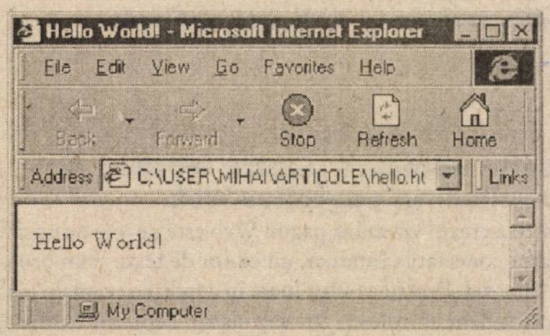
Un fișier HTML conține pe lângă simplul text care va apărea în pagină și anumite comenzi speciale, numite *tag*-uri. Fișierele HTML trebuie salvate întotdeauna cu extensia .htm sau .html dacă vă permite sistemul de operare.

Un *tag* este o comandă folosită pentru a indica modul de afișare a unei pagini Web. *Tag*-urile sunt cuvinte care încep cu simbolul "<" și se termină cu simbolul ">". Majoritatea *tag*-urilor au două părți, prima, numită *opening tag* indică unde începe fragmentul asupra căruia trebuie aplicată comanda iar a doua parte, *closing tag* arată sfârșitul fragmentului. *Closing tag*-urile se disting de *opening tag*-uri prin faptul că imediat după simbolul "<" apare

simbolul "/".

Fiecare fișier HTML trebuie să conțină *tag*-ul <HTML> care arată că respectivul fișier descrie într-adevăr o pagină Web. Deci orice astfel de fișier începe cu <HTML> și se încheie cu </HTML>. O pagină Web trebuie să aibă un antet, acesta fiind precizat în *tag*-ul <HEAD>; evident sfârșitul antetului este marcat de </HEAD>. De obicei singurul lucru conținut în antet este titlul paginii Web, titlul precizat între <TITLE> și </TITLE>. După antet urmează pagina Web propriu-zisă, descrierea ei fiind cuprinsă între <BODY> și </BODY>.

Primul exemplu va fi clasicul "Hello World!". O pagină Web care afișează un astfel de mesaj arată astfel:



Pentru a afișa acest mesaj într-o pagină Web aveți nevoie de următorul fișier HTML:

```
<HTML>
<HEAD>
<TITLE> Hello World!
</TITLE>
</HEAD>
<BODY>
Hello World!
</BODY>
</HTML>
```

Trebuie precizat că în HTML caracterele albe sunt folosite ca separatori. Astfel unul sau mai multe astfel de caractere consecutive sunt tratate ca un singur caracter alb. De exemplu pagina Web corespunzătoare fișierului HTML care urmează, este identică cu pagina Web anterioară:

```
<HTML> <HEAD> <TITLE> Hello World! </TITLE>
</HEAD><BODY>
Hello
                World!</BODY> </HTML>
```

Crearea de paragrafe și trecerea la o linie nouă

Așa cum am spus caracterele albe sunt practic ignorate de browserul care afișează o pagină Web, de aceea avem nevoie de comenzi speciale pentru a trece la o linie nouă. Cea mai folosită astfel de comandă este <BR> care nu face altceva decât să informeze browserul că textul care urmează va fi afișat pe o linie nouă. O altă comandă folosită pentru această operație este <HR>. Diferența dintre cele două *tag*-uri este că cea de-a doua desenează în plus o linie între cele două linii de text despărțite de <HR>.

Comanda <P> are ca efect tratarea textului care urmează după ea ca un paragraf nou. Singura diferență reală dintre <P> și <BR> este că <P> afișează în plus o linie vidă, menită să separe paragrafele.

Pentru comanda <P> *closing tag*-ul este opțional, un paragraf încheindu-se evident atunci când începe unul nou. Comenzile <HR> și <BR> nu au *closing tag*-uri.

În continuare vă prezentăm fișierul HTML pentru o pagină Web care ar conține o poezie în care fiecare vers trebuie afișat pe o linie, iar strofele reprezintă paragrafe:

```
<HTML>
<HEAD>
<TITLE> Sprites </TITLE>
</HEAD>
<BODY>
I once found a sprite,          <BR>
Like the Sun she was bright,    <BR>
One poor little creature        <BR>
Vanished in the future.         <P>
Eight centuries later <BR>
You will find my shelter,       <BR>
Only when your light  <BR>
Understands my night. <P>
Generations pass,       <BR>
And only when the grass         <BR>
Brings to you my fear <BR>
Real things disappear.<P>
I have lost my way      <BR>
Even if one day         <BR>
Looking at the ground <BR>
Another sprite I found.
</BODY>
</HTML>
```

Titluri

De obicei titlurile din documente apar mai mari și mai îngroșate decât restul cuvintelor. În HTML există mai multe *tag*-uri care arată că un anumit text va fi afișat ca un titlu. Acestea sunt <H1>, <H2>, <H3>, <H4> etc. Toate acestea au aceeași sintaxă, deci ajunge să ne ocupăm numai de una din ele. De exemplu prima dintre ele face ca textul cuprins între <H1> și </H1> să fie tratat ca un titlu. Titlul va fi afișat ca un nou paragraf și textul ce urmează va fi și el considerat un nou paragraf. Diferența dintre <H1>, <H2> și restul *tag*-urilor de acest tip este mărimea fontului cu care este afișat titlul. Pentru un titlu mare trebuie folosit <H1>, titlurile corespunzătoare celorlalte având dimensiuni tot mai mici. În practică nu se folosesc decât <H1>, <H2> și uneori <H3>, celelalte afișând titluri cu dimensiuni prea

mici pentru ca acestea să arate ca niște "adevărate" titluri.

Dacă dorim ca titlul poeziei anterioare să apară și în pagină, nu numai în bara de titlu, trebuie să adăugăm la începutul corpului: <H2> Sprites </H2>. Efectul poate fi observat în imaginea următoare:

Link-uri hypertext

Link-urile hypertext sunt legături între paginile Web care permit trecerea de la o pagină la alta. Faptul că unul sau mai multe cuvinte alăturate din text reprezintă un link este evidențiat de browser prin sublinierea acelor cuvinte și afișarea lor cu o altă culoare (albastru). Dacă veți muta cursorul deasupra cuvintelor ce marchează un astfel de link veți observa că acesta ia forma unei mâini. Apăsând butonul stâng al mouse-ului în acea zonă veți comanda browserului să treacă în zona corespunzătoare acestui link. Această zonă poate fi o altă pagină sau o anumită poziție din actuala pagină.

Pentru a crea un link la o altă pagină Web trebuie să folosiți *tag*-ul <A> care are următoarea sintaxă:

```
<A HREF=nume_pagină> text_tag </A>
```

Un exemplu de folosire a acestui *tag* este:

```
<HTML>
<HEAD>
<TITLE> Exemplu link </TITLE>
</HEAD>
<BODY>
...
pentru versiunea in limba engleza
<A HREF = "english.htm"> apasati aici! </A>
for the english version
<A HREF = "english.htm"> click here! </A>
...
</BODY>
</HTML>
```

Dacă fișierul ENGLISH.HTM nu se află în directorul curent sau nu este un fișier HTML atunci la selectarea unuia din cele două link-uri va fi afișat un mesaj de eroare și pagina curentă nu va fi schimbată.

Dacă se dorește ca legătura să se facă la un fișier care se află pe un alt calculator atunci trebuie precizată întreaga adresă URL (Uniform Resource Locators), adică numele serverului, urmat de calea până la fișierul dorit.

```
<A HREF="http://www.microsoft.com/guests/
welcome.html"> Apăsați aici! </A>
```

Pentru a face salturi la o anumită poziție din aceeași pagină sau la o poziție anume dintr-o altă pagină se folosește atributul *NAME* al *tag*-ului <A> pentru a marca acel loc: de exemplu <A NAME = "pozitia1"></A>. Pentru ca un link să facă un salt la această poziție din aceeași pagină trebuie ca în fișierul HTML să fie scris:

```
<A HREF="#pozitia1"> Apasati aici pentru
pozitia 1 </A>.
```

Pentru ca link-ul să facă un salt la o poziție oarecare din altă pagină comanda HTML corespunzătoare este:

```
<A HREF="altapagina.html#pozitia1"> Apasati
aici pentru pozitia1 din alta pagina </A>.
```

Se recomandă ca o pagină lungă să aibă în interiorul ei mai multe link-uri care să ducă la începutul paginii. Nu este permis să nu aveți un astfel de link într-un spațiu care depășește 2 ecrane.

Este posibilă crearea unor link-uri speciale care permit trimiterea unui mesaj e-mail la o anumită adresă. Pentru ca navigatorul să poată trimite un e-mail la o numită adresă linia HTML trebuie să arate astfel:

```
<A HREF="mailto:office@ginfo.ro"> Trimiteti
un mesaj pe adresa redactiei </A>.
```

Când acest link este selectat, browser-ul va deschide o fereastră care va permite trimiterea unui mesaj e-mail.

Formatarea textelor

Pentru a scrie texte în format bold folosiți *tag*ul <B>, iar pentru texte în format italic *tag*ul <I>. Textul cuprins între <B> și </B>, respectiv <I> și </I> va fi afișat ca bold sau italic. Puteți folosi și o combinație a acestor două *tag*uri astfel:

```
<B><I> TEXT </I></B>
```

Rezultatul va arăta cam așa: ***TEXT***. Formatul bold nu poate fi aplicat titlurilor deoarece ele sunt deja îngroșate.

Tag-ul <B> are ca echivalent *tag*ul <STRONG>, iar *tag*-ul <I> are ca echivalent *tag*-ul <EM>.

Tag-ul <SMALL> are ca efect micșorarea dimensiunii caracterelor asupra cărora are efect. *Tag*-ul pereche este <BIG> care mărește dimensiunea caracterelor.

Tag-ul <SUP> are ca efect afișarea textului corespunzător ca indice superior. *Tag*-ul pereche este <SUB>, acesta având ca efect afișarea caracterelor ca indice inferior.

Tag-ul <STRIKE> are ca efect desenarea unei linii peste text, iar *tag*-ul <U> determină sublinierea textului.

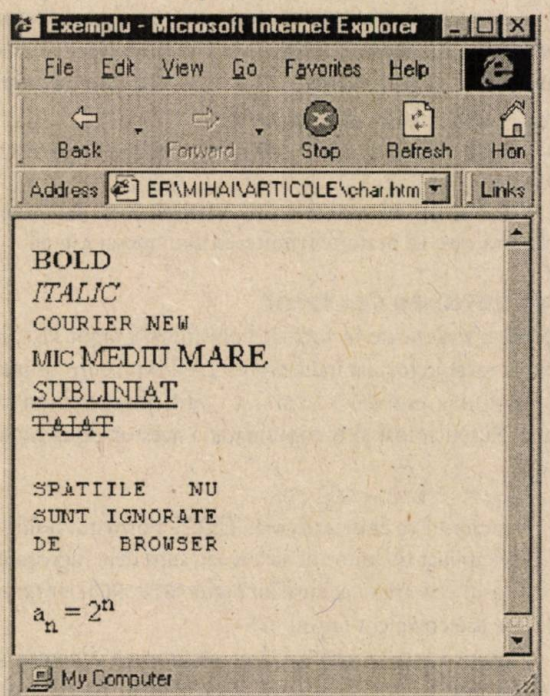
Tag-ul <TT> are ca efect schimbarea fontului pentru textul respectiv la Courier New.

Tag-ul <PRE> informează browserul că pentru textul respectiv caracterele albe nu mai sunt ignorate. Fontul este schimbat la Courier New.

Următorul fișier HTML ilustrează efectul acestor noi *tag*-uri.

```
<HTML>
<HEAD><TITLE> Exemplu </TITLE></HEAD>
<BODY>
<STRONG> BOLD </STRONG> <BR>
<EM>  ITALIC </EM> <BR>
<TT> COURIER NEW </TT> <BR>
<SMALL> MIC </SMALL> MEDIU <BIG> MARE </BIG>
<BR>
<U> SUBLINIAT </U> <BR>
<STRIKE> TAIAT </STRIKE>
<PRE>
SPATIILE          NU
SUNT IGNORATE
DE        BROWSER
</PRE>
a<SUB>n</SUB> = 2<SUP>n</SUP>
</BODY>
</HTML>
```

Pagina Web corespunzătoare ar arăta astfel:



Alinierea textelor

Unele *tag*-uri, după cum s-a văzut, permit specificarea unor opțiuni speciale. Pentru a alinia textul, mai multe *tag*-uri acceptă atributul ALIGN care permite alinierea textelor. Cel mai des folosit în acest scop este *tag*-ul <P>. Următoarele linii arată modul în care este folosit acest *tag* pentru ca textul din respectivul paragraf să fie aliniat la stânga, la dreapta sau centrat:

```
<P ALIGN = "center"> text </P>
<P ALIGN = "left"> text </P>
<P ALIGN = "right"> text </P>
```

Atributul ALIGN poate fi folosit si cu *tag*-urile <H1>, <H2> etc.

Pentru a seta alinierea textului pentru mai multe paragrafe sau titluri se poate folosi *tag*-ul <DIV> cu atributul ALIGN. *Tag*-ul <DIV> se numește *division* dar nu are nici un efect în versiunea HTML 3.2.

Mai există *tag*-ul <CENTER> pentru centrarea textelor, dar acesta a fost eliminat în mod oficial din standardul HTML și nu se recomandă folosirea lui deoarece versiuni viitoare ale browserelor s-ar putea să nu îl mai recunoască.

Setarea caracteristicilor fonturilor

Pentru a seta caracteristicile fonturilor se folosește *tag*-ul <FONT> care admite numeroase atribute.

Atributul SIZE definește mărimea caracterelor. El poate lua valori de la 1 (foarte mic) la 7 (foarte mare), valoarea normală fiind 3.

Atributul COLOR poate primi ca valori una din culorile standard: black, white, red, green, blue, yellow, aqua, fuchsia, gray, lime, maroon, purple, navy, olive, silver sau real.

Atributul FACE permite alegerea unui anumit font, font care va fi folosit pentru afișarea textului. Acest atribut va avea ca valoare o listă de fonturi separate prin ", ", iar browserul va alege din această listă primul font care există instalat pe mașina pe care este afișată pagina. De exemplu comanda:

```
<FONT FACE = "ShelleyAllegro BT, Comic Sans MS"> Font </FONT>
```

va avea ca efect: **Font** dacă fontul ShelleyAllegro BT este instalat și **Font** în caz contrar.

Se recomandă ca în listă să apară cel puțin un font care se află instalat pe majoritatea mașinilor, așa cum a fost Comic Sans MS în acest exemplu.

Comanda <BASEFONT> permite setarea mărimii fontului pentru tot textul care urmează. Comanda nu are un *closing tag* și nu permite alte atribute în afară de SIZE.

Majoritatea fonturilor includ caractere speciale cum ar fi caracterele speciale din mai multe limbi europene, printre care și româna. De exemplu pentru a scrie cuvântul: România în fișierul HTML trebuie să apară ceva de genul: Rom&#226nia.

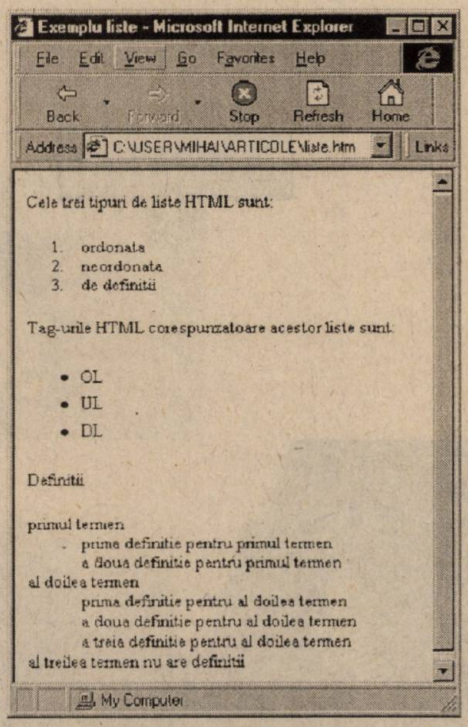
Aranjarea textului în liste

Listele HTML sunt de trei tipuri: liste ordonate, liste neordonate și liste de definiții. Pentru a informa browser-ul că urmează o listă ordonată se folosește *tag*-ul <OL>, pentru o listă neordonată *tag*-ul <UL>. Fiecare element al acestei liste trebuie precedat de *tag*-ul <LI>. Lista de definiții este indicată de *tag*-ul <DL>, fiecare element care trebuie definit este precedat de *tag*-ul <DT>, iar fiecare definiție este precedată de *tag*-ul <DD>. Următorul exemplu arată modul în care sunt folosite cele trei tipuri de liste:

```
<HTML>
<HEAD>
<TITLE> Exemplu liste </TITLE>
</HEAD>
<BODY>
Cele trei tipuri de liste HTML sunt:
<OL>
<LI> ordonata <LI> neordonata <LI> de definitii
</OL>
Tag-urile HTML corespunzatoare acestor liste
sunt:
<UL>
<LI> OL <LI> UL  <LI> DL
</UL>
Definitii
<DL>
<DT> primul termen
<DD> prima definitie pentru primul termen
<DD> a doua definitie pentru primul termen
<DT> al doilea termen
<DD> prima definitie pentru al doilea termen
<DD> a doua definitie pentru al doilea termen
<DD> a treia definitie pentru al doilea termen
<DT>  al treilea termen nu are definitii
</DL>
</BODY>
</HTML>
```

Pentru a crea liste imbricate nu trebuie decât să introduceți o listă în interiorul alteia. Browserul le va afișa automat imbricate. În cazul listelor neordonate browserul va schimba și tipul caracterului afișat la începutul rândului: astfel pentru primul nivel textul va fi precedat de o bulină plină, al doilea de un cerc iar pentru celelalte niveluri de imbricare de un pătrat plin.

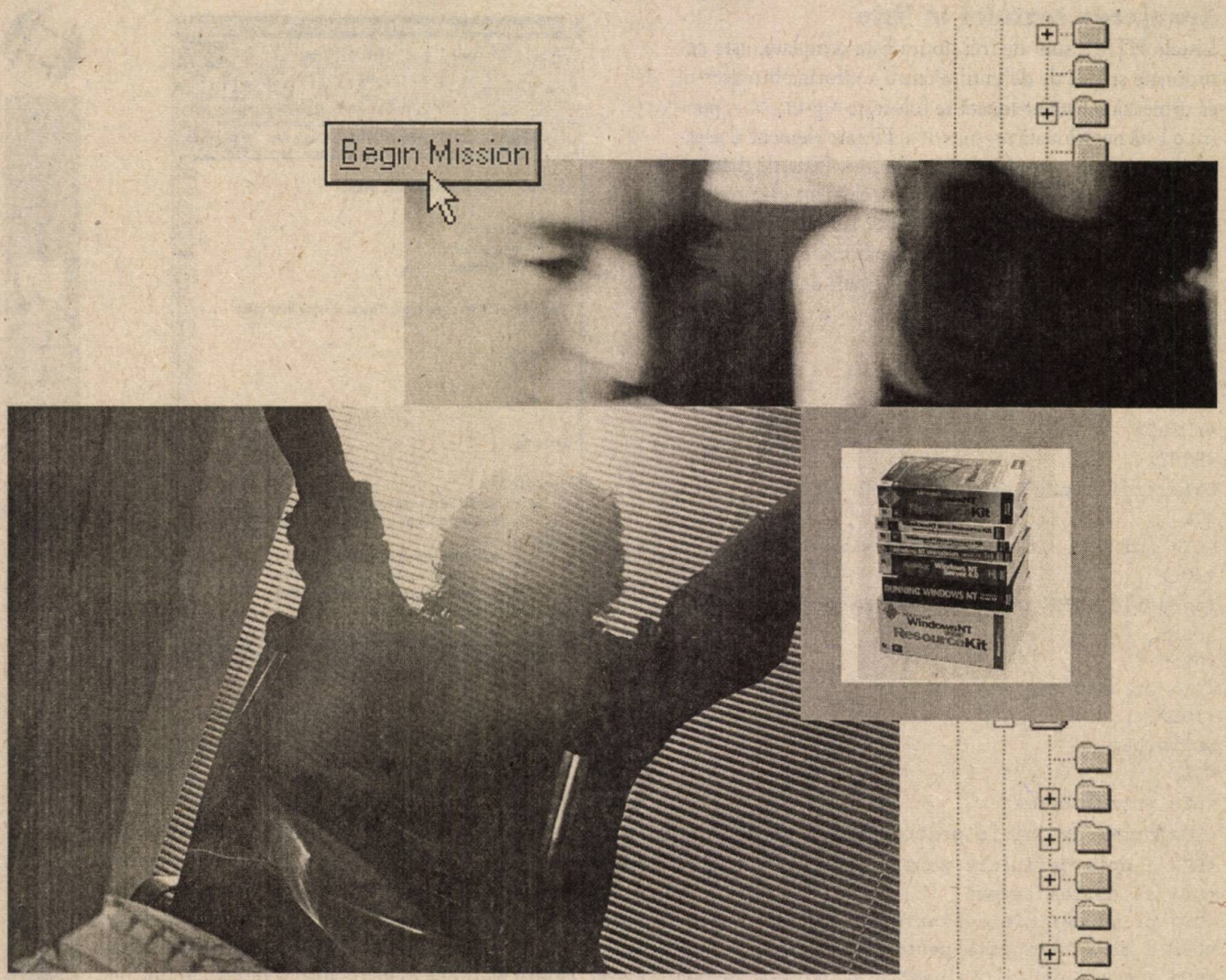
Efectul codului HTML în care sunt folosite trei tipuri de liste este următorul:



Pentru listele ordonate există posibilitatea de a preciza tipul elementelor care vor numerota lista cu ajutorul atributului *TYPE* al comenzii <OL>. Pentru TYPE="I" numerotarea se va face cu cifre romane; pentru TYPE="i" numerotarea se va face tot cu cifre romane, dar se vor folosi literele mici ale alfabetului; pentru TYPE="A" numerotarea se va face cu litere mari; pentru TYPE="a" numerotarea se va face cu litere mici. Poate fi precizată și valoarea de la care se va începe numerotarea cu ajutorul atributului START. Pentru START=6 numerotarea va începe, în funcție de valoarea atributului TYPE, cu 6, VI, vi, F, respectiv f. Există și posibilitatea de a preciza pentru un anumit element al listei valoarea care îl va preceda cu ajutorul atributului <VALUE> al *tag*-ului <LI>: <LI VALUE="12"> va face ca înaintea liniei să apară 12, XII, xii, L sau l, în funcție de valoarea atributului TYPE.

Pentru listele neordonate poate fi precizat tipul simbolului care va preceda elementele listei cu ajutorul atributului TYPE care poate lua valorile: "disc", "circle" sau "square". Poate fi parametru atât al *tag*-ului <UL>, cât și al *tag*-ului <LI>. În cel de al doilea caz va avea efect numai asupra liniei curente.

Mihai Scorțaru este student la Universitatea Tehnică din Cluj. Poate fi conectat pe e-mail: clara@trans.ro



Microsoft Press helps

you

create your own answers

WHERE DO YOU WANT to go today? ® Microsoft Press

COMPUTER PRESS AGORA

COMPUTER PRESS AGORA

Sucursala București, Str. Poterași 35/2, Tel.01-3309281, Fax: 01-3309285

MOTOARE de căutare

Dan Iancu

Volumul și diversitatea informației disponibile în Internet sunt enorme. Acest fapt reprezintă punctul forte al Rețelei dar, totodată, și principala slăbiciune. Cum putem găsi în neantul de informații paginile care ne interesează? Aceasta este o întrebare la care încearcă să răspundă acest articol. Pe scurt: cum să căutăm și mai ales, cum să începem navigarea în lumea virtuală...

După ce acționați iconul unui browser (program dedicat navigării pe Internet, deci îl vom numi în continuare "navigator") vă aflați în fața unei întrebări pe care și-o pune orice începător: "Unde vreau să mă duc?". Sfatul meu este ca, pentru început măcar, să intrați în spațiul dedicat scrierii de adrese URL (*Uniform Resource Locator*) și să scrieți adresa yahoo.com. Puteți pune aceeași adresă și în opțiunile navigatorului, ca pentru un timp să începeți de fiecare dată cu Yahoo. Acesta este numele celui mai mare motor de căutare în Internet. Puteți încerca și www.excite.com (chiar dacă nu este cel mai mare, a fost alegerea lui PC Magazine - 1 septembrie 1998).

Motoarele de căutare sunt realizate cu ajutorul unor baze de date (de exemplu Oracle) în care sunt stocate informații despre paginile de web (titlul, autorul, limba folosită, data creării, data ultimei modificări etc.) organizate în tabele de indecși. Aceste informații sunt accesate de programe puternice care folosesc algoritmi performanți de căutare pentru a obține rezultatul cât mai repede posibil.

Iată o listă a celor mai populare motoare de căutare:

Alta Vista	www.altavista.com
Yahoo	yahoo.com
Excite	www.excite.com
Lycos	www.lycos.com
WebSurfer	www.infohiway.com/way
Magellan	www.mckinley.com
Infoseek	www.infoseek.com
Webcrawler	webcrawler.com
HotBot	www.hotbot.com
Metacrawler	www.metacrawler.com

Notă: Metacrawler nu este un motor de căutare propriu-zis, deoarece utilizează serviciile altor motoare de căutare.

Glosar de termeni

Motor de căutare - Pagină de web sau program care plecând de la unul sau mai multe cuvinte cheie îți oferă o listă de adrese pe Internet unde acel cuvânt cheie (sau frază) poate fi găsit. În engleză este *search engine*.

Search, Find (Caută). Îl găsiți pe butonul alăturat locului unde scrieți cuvântul cheie.

Keyword (Cuvânt cheie). Cuvintele cheie sunt desemnate de cel care a construit pagina de web respectivă pentru a facilita căutarea după criterii dorite de el. De exemplu, dacă vă construiți o pagină personală, când vă înregistrați într-un motor de căutare va trebui să completați de obicei o formă în care găsiți și un câmp destinat cuvintelor cheie. Acolo veți scrie acele cuvinte care credeți că descriu cel mai bine pagina voastră și în același timp vor face ca o mare parte dintre cei care navighează pe web să vă găsească. În același timp, cuvântul cheie poate fi găsit de motorul de căutare doar ca simplu cuvânt în pagină.

Frază. O secvență de cuvinte scrise între ghilimele. Fraza se comportă ca un singur cuvânt cheie, adică fiecare pagină din listă va conține toate cuvintele din frază.

Expresie booleană. O secvență de cuvinte cheie unite prin operatorii logici *AND, OR, NOT, NEAR*. În caseta "Operatori logici" sunt prezentate funcțiile acestora.

Ierarhie. Un arbore de subiecte generale, care dă posibilitatea celui care caută să găsească ceea ce-l interesează, știind doar informații lipsite de specific. Aceasta reprezintă o modalitate de căutare în care vă interesează nu numai o anumită pagină, ci permite vizualizarea tuturor paginilor care sunt legate la acel subiect general.

Categorie. Tipul de subiect abordat de pagina de web. Căutarea pe web se face nu numai după cuvinte cheie ci și după categorii (vezi *ierarhie*). De exemplu, în Yahoo sub câmpul pentru căutarea după cuvinte cheie, o ierarhie de

Operatorii logici

Operator	Simbol	Cum acționează:
AND	&	Ambii termeni ai expresiei trebuie să fie găsiți în pagina respectivă. De exemplu expresia ***agora AND line*** va avea ca rezultat toate paginile care conțin ambele cuvinte.
OR	|	Măcar unul dintre cuvinte trebuie să existe în pagină. De exemplu, expresia ***agora OR line*** va avea ca rezultat toate paginile care conțin măcar unul din cele două cuvinte.
NOT	!	Este un operator unar și este aplicat doar termenului care îl urmează. El nu poate fi folosit singular. De exemplu ***agora AND NOT line*** va avea ca rezultat toate paginile care conțin cuvântul ***agora***, dar nu conțin cuvântul ***line***. Expresia ***agora NOT line*** nu este acceptată.
NEAR	~	Găsește toate paginile care conțin ambele cuvinte la o distanță de cel puțin 10 cuvinte între ele.

categorii generale. Mergând mai departe pe una din aceste categorii veți găsi o listă de subcategorii și așa mai departe. De exemplu: Home : Science : Computer Science : Algorithms (http://dir.yahoo.com/Science/Computer_ Science/Algorithms/Sorting/) este o ierarhie unde Science este o subcategorie și la Computer & Science (http:// www.yahoo.com/Computers/) deși, conform adresei date ierarhia nu este respectată. În cazul înregistrării paginii voastre de web într-un motor de căutare există posibilitatea de a fi trecută la mai multe categorii (de obicei trei) pentru a vă lăsa o anumită elasticitate în descriere. Categoria sau subcategoria poate fi și un cuvânt cheie.

Meta-tag. Comenzi în limbajul HTML în care sunt descrise cât mai multe informații despre pagina de web în care se află. Nu toate motoarele de căutare iau în considerare meta-tagurile (AltaVista da, Excite nu), dar folosirea lor este importantă pentru că dau posibilitatea persoanei interesate - în cazul înscrierii în mai multe astfel de motoare - să ajungă la pagina respectivă.

Meta-tag Builder. O pagina de web care după completarea și trimiterea (*send*) unui formular (forme) vă trimite la adresa voastră de e-mail o secvență de meta-tag-uri care trebuie inserată în codul de HTML al paginii respective.

Puncte de start

În afară de ierarhiile furnizate de cele mai importante motoare de căutare, există numeroase pagini Web care furnizează colecții de trimiteri (link-uri) structurate tematic. Aceste pagini sunt numite de regulă *puncte de start*. Iată câteva dintre ele:

WWW Virtual Library (http://vlib.org) - O vastă colecție de adrese. Ierarhia este bazată pe 14 categorii și nu este foarte adâncă, dar trimiterile sunt îndreptate către puncte de start tematice, atent selecționate.

LookSmart (http://www.looksmart.com/) - Un punct de start foarte atractiv prin interfață și prin structurarea tematicii. Se ajunge la 5-6 niveluri în adâncime.

Miningco.net (http://home.miningco.com/) - Un punct de start care reunește circa 500 de ghiduri tematice, totalizând mii de link-uri.

Component-Oriented Programming (www.math.tau.ac.il/~guy/COP/) - O colecție de link-uri tematice, structurate și ținute la zi de Guy Laden la Universitatea din Tel-Aviv.

Cum căutăm după cuvinte cheie

Cel mai simplu e să scriem cuvântul dorit și să apăsăm pe *search*. Vom primi o listă bogată cu multe poziții. De exemplu am căutat cu Yahoo cuvântul "agora" și am primit peste 40.000 de adrese. Când am scris "agora ON line" rezultatul a fost mult mai... rău! Peste 200.000. Atunci mi-a venit ideea salvatoare: să pun cuvintele între ghilimele și rezultatul a fost în sfârșit cel dorit: 21 (206 în AltaVista!) de poziții din care câteva erau din revista noastră.

Reguli de căutare după cuvinte cheie:

Căutarea, de obicei nu ține cont de majuscule sau de minuscule, adică nu este ceea ce se numește "*case sensitive*". Programul transformă majusculele în minuscule, chiar dacă textul este scris cu majuscule. De exemplu "agora ON line" a fost căutat cu "agora on line". Doar AltaVista face diferența. În cazul în care nu sunteți siguri de cum trebuie scris, folosiți minuscule pentru a primi cât mai multe adrese.

Folosiți cât mai multe cuvinte cheie, pentru că motorul de căutare va începe prin a căuta paginile care au cât mai multe din cuvintele cheie. În cazul în care doriți o anumită frază, scrieți-o între ghilimele (").

Folosiți expresii booleene pentru a da informații suplimentare motorului de căutare. O expresie booleană poate fi formalizată astfel:

<a><operator><b>, unde *a* și *b* pot fi cuvinte, fraze sau chiar expresii booleene. Expresiile booleene pot conține și paranteze pentru a putea grupa diverși termeni după criterii logice. De exemplu, o expresie de acest fel este: "agora

on line" *AND* (editorial *OR* "don pedrito") *AND NOT* (cuprins *OR* galeria).

În căutările simple puteți folosi și semnele + și -. De exemplu *expresia +agora +on +line -editorial* va căuta paginile care conțin cuvintele *agora*, *on* și *line*, fără cuvântul *editorial*.

Simbolul "*" plasat la sfârșitul unui cuvânt va produce o căutare a tuturor cuvintelor care au diferite încheieri. Este folositor pentru căutarea familiei de cuvinte. De exemplu, dacă vrem să căutăm cuvintele *poem, poems, poetry, poezie, poet* putem căuta *poe**.

Orice motor de căutare are o pagină sau mai multe în care găsiți indicații utile despre cum să vă îmbunătățiți căutarea. Pot fi folosite și alte criterii (de exemplu AltaVista are posibilitatea de a căuta după cuvinte speciale de tip *url, imagine, text* etc. - **www. altavista.com /av/content/help.htm**) sau puteți rafina căutarea după data când a fost făcută pagina respectivă. Înainte de a purcede la o căutare, încercați să citiți cu atenție indicațiile date în paginile de "help". Nu toate motoarele de căutare au aceleași criterii și posibilități.

Căutare după ierarhie

Orice motor de căutare are și posibilitatea de a căuta după categorii în ierarhii definite după criterii proprii. Acest tip de căutare se face destul de lejer (abstracție făcând de timpul de așteptare) prin simpla apăsare pe categoria dorită. Vom primi o subierarhie și o listă de adrese la unele dintre motoare, cum ar fi Yahoo) care sunt legate direct la categoria accesată. Putem continua astfel până la terminarea subîmpărțirilor sau până la găsirea paginilor dorite.

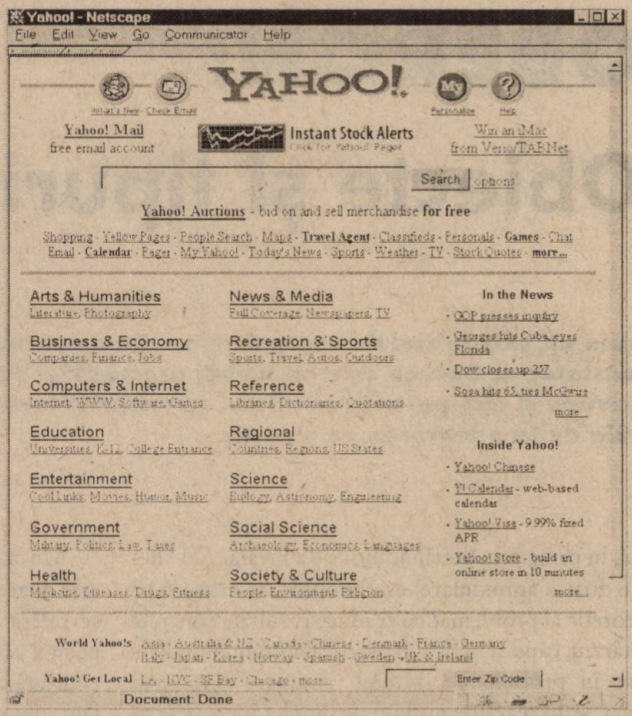
P.S.

Despre motoare de căutare românești citiți în revista "agora ON line" la *Coloana Infinitului*, numărul din octombrie, 1998.

Dan Iancu este redactor șef al revistei "agora ON line" (www.aol.ro). Poate fi contactat la adresa diancu@agora.ro.

Categorii și ierarhii

Motorul de căutare Yahoo oferă o ierarhie compusă din 14 categorii principale fiecare dintre acestea constituind rădăcina unei subierarhii.



Exemplu de subierarhie în Yahoo pentru categoria „Internet:"

Business and Economics (85)
Chat (893)
Child Safety (16)
Commercial Services@
Commercial Software@
Communications and Networking@
Companies@
Conferences and Events (88)
Connectivity (9)
Courses (13)
Directory Services (14)
Domain Registration (100)
Electronic Mail@
Employment (290)
FTP Sites (29)
Games@
Gopher (26)
History (34)
Humor@
Indices to Web Documents@
Information and Documentation (178)
Interesting Devices Connected to the Net (739)
Internet 2 - Abilene Project (11)
Internet Fax Server (7)
Internet Phone (5)
Intranet@
Law@
Magazines@
Mailing Lists (59)
Maps (15)
Network Topology (16)
News and Media (178)
Online Teaching and Learning@
Organizations (74)
Policy (103)
Resources (42)
Searching the Net (30)
Software@
Statistics and Demographics (64)
Usenet (247)
Usenet Groups (5)
User Groups (34)
World Wide Web (3004)

OBERON:
Obiecte și tipuri

Mircea Sârbu

Dacă ați citit articolul despre Oberon din numărul precedent al revistei, ați constatat că Oberon este un limbaj asemănător Pascalului, mai simplu în unele privințe și mai puternic în altele. Adevăratul farmec al limbajului începe însă abia de la orientarea pe obiecte.

Există, în principiu, două moduri de a învăța. Cel "deductiv" (e doar o aproximare) este ordonat, se bazează pe studiul teoretic al problematicii și aduce rezultate concrete abia mai târziu, când se aplică în practică noțiunile studiate și înțelese în profunzime. Modul "inductiv" este mai dezordonat, se bazează pe noțiuni incomplet înțelese și pe practici lipsite de rigoare. Altfel spus: "după ureche". Cu toate acestea, construcția – oricât de șchioapă pare – poate fi foarte ușor stabilizată cu ajutorul unei cărți bune (de pildă), care poate pune în ordinea corectă noțiunile disparate acumulate și exersate deja.

În privința tehnologiei obiectuale (în engleză *Object-Oriented*, abreviat adesea *OO*) se pare că nu aveți de ales: deoarece în liceu nu se studiază iar în reviste se tot vorbește, sunteți obligați să adoptați metoda "inductivă". Deja v-ați întâlnit cu "obiectele" în Java, iar în acest număr le întâlniți și într-o variantă de Pascal folosită în Delphi.

Tehnologia obiectuală

Dacă, totuși, ați vrea să abordați problema frontal și să studiați domeniul obiectual în mod sistematic, s-ar putea să fiți repede descurajați sau dezamăgiți. Cartea considerată de referință în domeniu (Bertrand Meyer, *Object-Oriented Software Construction*, Prentice Hall, 1997) are aproape 1300 de pagini și este inabordabilă fără o pregătire prealabilă. În plus, nu există un model obiectual unanim acceptat, fiecare limbaj obiectual se bazează pe o altă abordare și folosește o altă terminologie. Simptomatic este și faptul că în limba română nici măcar nu există un consens privind traducerea termenului *object-oriented* – se spune "orientat pe/spre/către obiecte", sau doar "orientat obiect", uneori "pe bază de obiecte", sau (varianta pe care o prefer eu) "obiectual".

Un alt aspect este faptul că programarea obiectuală este doar o latură a tehnologiei obiectuale, tehnologie care se referă la întreg ciclul de viață a unui program. Este posibil ca întreaga analiză ce stă la baza programului să fie realizată cu metode obiectuale iar implementarea să se facă fără obiecte. Sau invers. Mai mult, este posibil să se programeze obiectual (sau, măcar, în stil obiectual) în limbaje care nu sunt orientate pe obiecte.

Pe de altă parte, în lumea obiectuală există un conflict de idei, cu unele accente de fanatism, între "puriști" (sau "ortodocși") și "pragmatici". În mod oarecum analog, există limbaje obiectuale "pure" și limbaje clasice cu extensii obiectuale (limbaje zise "hibride"), precum și toată gama de variante intermediare.

Când a creat limbajul Oberon, profesorul Wirth și-a dorit să evite cu totul "șocul cultural" pe care programatorii erei pre-obiectuale îl resimt în fața noii tehnologii. În consecință, a evitat în totalitate jargonul specific și a abordat aspectele obiectuale ca simple extensii ale noțiunilor clasice. Din acest motiv, Oberon este probabil calea cea mai ușoară către programarea orientată pe obiecte. Însă, pentru ca noțiunile pe care vi le prezint în rândurile următoare să fie utile și în cazul altor limbaje, voi utiliza un limbaj "mixt" (adică voi specifica și termenii corespunzători din "jargonul" obiectual).

Ce știam deja...

Ideea de la care pleacă tehnologia obiectuală este, în esență, simplă: structurile de date și prelucrările care se efectuează asupra acestora sunt în mod intrinsec legate, deci ar fi mai bine dacă această legătură ar fi explicit exprimată. Intuitiv, o structură de date împreună cu prelucrările care-i sunt specifice formează un "obiect".

Avantajele introducerii obiectelor în activitățile legate de realizarea aplicațiilor sunt multiple. Iată doar câteva dintre ele:

• Proiectarea aplicațiilor se ordonează în mod firesc în jurul acestor obiecte;

• Deoarece prelucrările sunt legate de structurile de date asupra cărora sunt definite, se elimină o sursă majoră de erori;

• Obiectele fiind relativ autonome, odată ce au fost proiectate pot fi utilizate și în alte aplicații. Reutilizarea reprezintă un factor important de creștere a productivității.

Să încercăm în cele ce urmează să limităm cadrul discuției la programare (până acum m-am referit la proiectarea aplicațiilor în general) și să ilustrăm ideea printr-un scurt exemplu.

Să considerăm reprezentarea într-un program a unei persoane. Pentru simplitate, putem considera că ne interesează doar numele, data nașterii și localitatea de domiciliu. O reprezentare posibilă este următoarea:

```
TYPE
   Pers = POINTER TO PersDesc;
   PersDesc = RECORD
      lname, fname, address: ARRAY OF CHAR;
      born: Dates.Date
   END;
```

Desigur, în Pascal sintaxa ar fi puțin diferită. Observăm că este nevoie să importăm modulul *Dates*, care cuprinde reprezentarea datei calendaristice. Din mai multe motive (care se vor evidenția mai târziu), se preferă reprezentarea dinamică, prin pointer. O persoană va fi așadar reprezentată printr-o variabilă de tip *Pers*.

Important în acest context este faptul că există un set de operații care se pot defini asupra tuturor persoanelor. Desigur, ne interesează doar cele ce au sens și care sunt relevante pentru aplicația pe care o proiectăm. Implementarea acestora se face prin proceduri. Iată antetul câtorva dintre ele:

```
PROCEDURE Init* (IN lname, fname, address:
      ARRAY OF CHAR; born: Dates.Date): Pers;
```

Funcția returnează o nouă variabilă de tip *Pers*, inițializată cu valorile furnizate ca parametri.

```
PROCEDURE Update* (VAR p: Pers;
      IN lname, fname, address: ARRAY OF CHAR);
```

Procedura actualizează anumite date relative la o persoană. Implementarea poate recurge la convenția că pentru câmpurile care rămân neschimbate se furnizează ca parametru șirul vid (""). Schimbarea datei nașterii unei persoane nu are sens.

```
PROCEDURE Age* (p: Pers): INTEGER;
```

O funcție care returnează vârsta unei persoane, exprimată în ani.

```
PROCEDURE Mature* (p: Pers): BOOLEAN;
```

O funcție care returnează *TRUE* dacă persoana este majoră, altfel *FALSE*.

```
PROCEDURE GetName* (p: Pers; OUT n: ARRAY OF CHAR);
```

O procedură care furnizează în variabila *n* numele complet al persoanei (întâi prenumele și apoi numele).

Se mai pot imagina și altele dar, pentru simplitate, ne oprim aici. Putem să considerăm (deocamdată) că o variabilă de tip *Pers* împreună cu aceste proceduri reprezintă un ***obiect***. Valorile câmpurilor reprezintă ***starea*** obiectului (*status*), în timp ce procedurile reprezintă ***comportamentul*** obiectului (*behaviour*).

Este evident că toate variabilele de acest tip vor avea aceleași câmpuri și același comportament, dar pot avea stări diferite. Chiar dacă două obiecte au aceeași stare (valorile câmpurilor sunt egale), ele sunt totuși diferite, deci au o *identitate* proprie.

În limbaj obiectual, tipul *Pers* se numește ***clasă*** iar variabilele de tip *Pers* se cheamă ***instanțe*** ale clasei *Pers*. Clasa este un model, un tipar pe baza căruia sunt create obiecte. Comportamentul este, de fapt, specific clasei și nu obiectului propriu-zis.

Diferențele teoretice dintre o clasă și un tip de date sunt subtile și depășesc cadrul acestei prezentări. Din punctul de vedere al limbajului Oberon, clasele și tipurile de date de tip articol (*record*) sunt echivalente (desigur, este vorba și de pointerii corespunzători).

Ați observat cu siguranță că încă nu am pătruns efectiv în zona programării orientate pe obiecte, deoarece legătura dintre structura de date și procedurile care o procesează nu este *explicită*. Vom ajunge cu timpul și aici. Deocamdată putem pune în evidență doar o posibilitate de a programa în stil obiectual. Pentru aceasta va trebui să ne conformăm unui principiu foarte important pentru programarea obiectuală: încapsularea.

Încapsularea

Dacă vă voi spune că un sinonim pentru încapsulare este "ascunderea informațiilor" veți prinde din zbor ideea: trebuie să controlez accesul la obiecte.

În primul rând, trebuie să limităm procesările asupra structurii de date la cele pe care le considerăm necesare și sigure. În virtutea încapsulării, actualizarea stării unui obiect (de tip *Pers*, să zicem) nu trebuie să fie posibilă decât prin procedurile anume desemnate (în cazul nostru, *Update*). Pentru aceasta trebuie să avem posibilitatea să definim o ***interfață*** a obiectului, care să fie publică și obligatorie. Accesul total la starea unui obiect trebuie să rămână apanajul exclusiv al implementării acestei interfețe.

În felul acesta, un obiect devine pentru lumea înconjurătoare un fel de "cutie neagră". Din exterior nu este accesibilă decât interfața, prin care clientul poate solicita

Aproape obiectual

```
MODULE MirPers;

   IMPORT Dates;

   TYPE
      Pers* = POINTER TO RECORD
         lname, fname, address-: ARRAY 80 OF CHAR;
         born: Dates.Date
      END;

   PROCEDURE Age* (p: Pers): INTEGER;
      VAR today: Dates.Date;
   BEGIN
      Dates.GetDate(today); RETURN today.year - p.born.year
   END Age;

   PROCEDURE Mature* (p: Pers): BOOLEAN;
   BEGIN
      RETURN Age(p) >= 18
   END Mature;

   PROCEDURE GetName* (p: Pers; OUT n: ARRAY OF CHAR);
   BEGIN
      n := p.fname + ' ' + p.lname
   END GetName;

   PROCEDURE Update* (VAR p: Pers; IN lname, fname, address:
                                          ARRAY OF CHAR);
   BEGIN
      IF lname # "" THEN p.lname := lnameS END;
      IF fname # "" THEN p.fname := fnameS END;
      IF address # "" THEN p.address := addressS END
   END Update;

   PROCEDURE Init* (IN lname, fname, address: ARRAY OF CHAR;
                                    born: Dates.Date): Pers;
      VAR p: Pers;
   BEGIN
      NEW(p); p.born := born;
      Update(p, lname, fname, address);
      RETURN p
   END Init;

END MirPers.

MODULE MirPersTest;

   IMPORT Dates, Out, P := MirPers;

   PROCEDURE WritePers (p: P.Pers);
      VAR s: ARRAY 80 OF CHAR;
   BEGIN
      Out.Ln; P.GetName(p, s); Out.String(s); Out.Ln;
      Out.String("varsta: "); Out.Int(P.Age(p), 0); Out.Ln;
      Out.String("Adresa: " + p.address); Out.Ln
   END WritePers;

   PROCEDURE Test*;
      VAR p: P.Pers; d: Dates.Date;
   BEGIN
      d.year := 1980; d.month := 6; d.day := 21;
      p := P.Init("Icsulescu", "John", "Chicago", d);
      IF P.Mature(p) THEN WritePers(p) END
   END Test;

END MirPersTest.
```

anumite procesări (de regulă inițializări, actualizări și interogări). Modul în care acestea sunt efectiv realizate este treaba obiectului.

Avantajele încapsulării sunt evidente:

• Programele sunt mai sigure, deoarece invarianții specifici unui obiect nu pot fi afectați din exterior. Erorile pot fi astfel mult mai ușor localizate și corectate.

• Programele devin mult mai ușor de înțeles și de întreținut, deoarece detaliile de implementare sunt ascunse.

• Schimbarea implementării unei clase de obiecte nu afectează restul aplicației, în condițiile păstrării interfeței.

Încapsularea este realizată în Oberon prin intermediul modulelor și a marcării explicite a elementelor care sunt exportate (ele reprezintă, de fapt, interfața). În caseta "Aproape obiectual" puteți vedea sursa unui modul (*MirPers*) care implementează tipul *Pers* și comportarea acestuia, precum și un modul (*MirPersTest*) care importă modulul MirPers și folosește tipul *Pers*. Iată cum arată interfața modulului *MirPers* în prezentarea browser-ului de interfețe din mediul BlackBox Component Builder:

```
DEFINITION MirPers;
   IMPORT Dates;
   TYPE
      Pers = POINTER TO RECORD
         address-: ARRAY 80 OF CHAR
      END;
   PROCEDURE Age (p: Pers): INTEGER;
   PROCEDURE GetName (p: Pers; OUT n: ARRAY OF CHAR);
   PROCEDURE Init (VAR p: Pers; IN lname, fname, address:
                        ARRAY OF CHAR; born: Dates.Date);
   PROCEDURE Mature (p: Pers): BOOLEAN;
   PROCEDURE Update (VAR p: Pers; IN lname, fname, address:
                        ARRAY OF CHAR);
END MirPers.
```

Observații:

• Implementarea este realizată în Component Pascal, o extensie a limbajului Oberon-2. Această extensie permite declararea unui tip pointer în acest mod. În limbajul standard, declarația trebuie făcută prin intermediul unui tip static (în text am folosit *PersDesc*).

• Declararea parametrilor transmiși prin referință ca *IN* sau *OUT* este o extensie Component Pascal care permite un control mai strict al eventualelor efecte secundare.

• Declarația de import *P := MirPers* ne permite să prefixăm cu *P* referirile la modulul *MirPers* (pentru comoditate).

• Câmpul *address* este marcat cu semnul "-" (minus), ceea ce înseamnă că este accesibil din exterior doar pentru citire (*read only*).

• Operatorul "$" aplicat unei variabile de tip *ARRAY OF CHAR* extrage din variabilă doar valoarea string-ului (altfel este vorba de o operație cu tablouri de caractere).

Este interesant de remarcat că utilizatorii obiectelor nici măcar nu știu dacă o anumită informație de stare a

obiectului (de pildă vârsta) este stocată sau este calculată. Expunerea câmpului *address* (chiar limitată, doar pentru citire) este considerată de către puriștii OO ca o derogare la principiul încapsulării...

În fine, este evident că se poate programa în modul acesta "aproape obiectual" în orice limbaj, inclusiv în Pascal. Absența unor facilități de limitare a accesului poate fi compensată printr-o disciplină strictă și printr-o documentare detaliată a interfețelor. Efectele în plan calitativ nu vor întârzia să apară.

Articole extensibile

Calea pe care Oberon a abordat programarea orientată pe obiecte a fost deschisă de posibilitatea de a extinde tipurile declarate ca *RECORD*. O inovație minoră în aparență, dar esențială în esență. Să considerăm declarațiile următoare (în modulul *MirPers*):

```
TYPE
   Pers = POINTER TO EXTENSIBLE RECORD
      lname, fname, address*: ARRAY OF CHAR;
      born: Dates.Date
   END;

   Ang = POINTER TO RECORD (Pers)
      id-, income: INTEGER
   END;
```

Noul tip *Ang* este o ***extensie*** a tipului *Pers*. O primă consecință a acestui fapt este că tipul *Ang* ***moștenește*** toate câmpurile tipului de bază (*Pers*), la care se adaugă câmpurile sale proprii. Extinderea unui tip reprezintă, în general, o specializare a tipului de bază și exprimă o relație de genul "*Ang* este un fel de *Pers*". Într-adevăr, prin tipul *Ang* ne propunem să reprezentăm angajații unui firme. Desigur, aceștia sunt tot niște persoane, dar au în plus o marcă (*id*) și un salariu (*income*).

Observație: Declarația *EXTENSIBLE* este o extensie specifică Component Pascal. Standardul Oberon-2 nu prevede astfel de atribute pentru tipuri.

Faptul că definiția este mai scurtă în acest fel este un câștig absolut derizoriu. Marele câștig pe care-l obținem este faptul că putem folosi tipul extins oriunde putem folosi tipul de bază (cu o excepție pe care o vom trata la momentul potrivit).

Pentru a putea folosi în continuare obiecte de tip *Ang*, avem nevoie de un mecanism de creare și inițializare. O variantă comună este să definim (tot în modulul *MirPers*) o funcție *AInit*, într-un mod asemănător celei pentru persoane:

```
PROCEDURE AInit* (IN lname, fname, address:
      ARRAY OF CHAR; born: Dates.Date;
      id, income: INTEGER): Ang;
   VAR p: Ang;
BEGIN
   NEW(p); p.born := born;
   p.lname := lname$; p.fname := fname$;
   p.address := address$;
   p.id := id; p.income := income;
   RETURN p
END AInit;
```

Din acest moment, în orice modul care importă *MirPers* (se mai spune "modul client") putem să declarăm variabile de tip *Ang*, să le inițializăm și să le folosim la fel ca pe variabilele de tip *Pers*. De pildă, putem modifica procedura *Test* din modulul *MirPersTest* astfel:

```
PROCEDURE Test*;
   VAR a: P.Ang; p: P.Pers; d: Dates.Date;
BEGIN
   d.year := 1980; d.month := 6; d.day := 21;
   a := P.AInit("Icsulescu", "John", "Chicago", d, 9154, 2500);
   IF P.Mature(a) THEN WritePers(a) END;

   (* etc. *)

END Test;
```

Rezultatul executării acestei comenzi (în general, o procedură fără argumente exportată de un modul se numește "comandă") este previzibil. În fereastra *Log* se va afișa:

```
John Icsulescu
varsta: 18
Adresa: Chicago
```

Deci au fost executate fără probleme procedurile *MirPersTest.WritePers* și *MirPers.Mature* (care la rândul lor au apelat procedurile *Age* și *GetName* din modulul *MirPers*) deși sunt definite pentru tipul *Pers*. Deoarece la apelarea unei proceduri are loc un set de atribuiri (variabilelor definite ca parametri formali li se atribuie valorile furnizate ca parametri actuali) înseamnă că o atribuire de genul *p := a* (unde *p* este *Pers* și *a* este Ang) este legală. Într-adevăr, putem încerca această atribuire în procedura *Test* și totul este ok (atât la compilare cât și la execuție).

Explicația este că un obiect (o variabilă de tip articol sau pointer la articol) are un ***tip static*** și un ***tip dinamic***. Tipul static este cel declarat explicit în program (de pildă *Ang* pentru *a* și respectiv *Pers* pentru *p*). Tipul dinamic este cel pe care obiectul îl are la momentul execuției. Acesta poate fi același cu tipul său static sau poate fi o extensie a acestuia.

După atribuirea *p := a* variabila *p* va avea tipul static *Pers* iar tipul dinamic *Ang*. Este important de subliniat că deși acum *p* dispune și de câmpurile *id* și *income*, ele nu pot fi accesate (în forma *p.id* și respectiv *p.income*)

Obiectual timpuriu

```
MODULE MirPers;
   IMPORT Dates;
   TYPE
      Message* = EXTENSIBLE RECORD END;
      GetAgeMsg* = RECORD (Message) age*: INTEGER END;
      GetNameMsg* = RECORD (Message) name*:
                              ARRAY 80 OF CHAR END;
      UpdateMsg* = RECORD (Message)
         lname*, fname*, address*: ARRAY 80 OF CHAR
      END;
      Pers* = POINTER TO RECORD
         lname, fname, address-: ARRAY 80 OF CHAR;
         born: Dates.Date;
         handle*: PROCEDURE (p: Pers; VAR m: Message)
      END;
   PROCEDURE Age (p: Pers): INTEGER;
      VAR today: Dates.Date;
   BEGIN
      Dates.GetDate(today); RETURN today.year - p.born.year
   END Age;
   PROCEDURE GetName (p: Pers; OUT n: ARRAY OF CHAR);
   BEGIN n := p.fname + ' ' + p.lname
   END GetName;
   PROCEDURE Update (VAR p: Pers; IN lname, fname, address:
                                       ARRAY OF CHAR);
   BEGIN
      IF lname # "" THEN p.lname := lname$ END;
      IF fname # "" THEN p.fname := fname$ END;
      IF address # "" THEN p.address := address$ END
   END Update;
   PROCEDURE HandlePers (p: Pers; VAR m: Message);
   BEGIN
    WITH m: GetAgeMsg DO m.age := Age(p)
    | m: GetNameMsg DO GetName(p, m.name)
    | m:UpdateMsg DO Update(p, m.lname, m.fname, m.address)
    ELSE (* ignora mesajul *)
    END
   END HandlePers;
   PROCEDURE Init* (IN lname, fname, address: ARRAY OF CHAR;
                                    born: Dates.Date): Pers;
      VAR p: Pers;
   BEGIN
      NEW(p); p.born := born; p.handle := HandlePers;
      Update(p, lname, fname, address); RETURN p
   END Init;
END MirPers2.

MODULE MirPers2Test;
   IMPORT Dates, Out, P := MirPers;
   PROCEDURE Test*;
    VAR p: P.Pers; d: Dates.Date;
        getAge: P.GetAgeMsg; getName: P.GetNameMsg;
        update: P.UpdateMsg;
   BEGIN
    d.year := 1980; d.month := 6; d.day := 21;
    p := P.Init("Icsulescu", "Ioana", "Chicago", d);
    p.handle(p, getAge);
    IF getAge.age >= 18 THEN
       p.handle(p, getName); Out.String(getName.name); Out.Ln
    END;
    update.lname := "Igrecescu"; update.fname := "";
   update.address := "";
    p.handle(p, update);
       p.handle(p, getName); Out.String(getName.name); Out.Ln
   END Test;

END MirPersTest.
```

deoarece compilatorul nu cunoaște decât tipul static al variabilei *p* (în acest caz *Pers*), care nu dispune de aceste câmpuri.

Să mai notăm că o atribuire în sens invers (*a := p*) nu este legală. Regula este că tipul de bază este compatibil cu extensiile sale (numite și subtipuri), dar nu și invers.

Este momentul să semnalăm și excepția de la compatibilitate pe care o semnalam mai înainte: parametrii formali declarați *VAR*, *IN* sau *OUT* nu acceptă decât variabile al căror tip static este exact cel specificat pentru parametrul formal. Acesta este motivul pentru care procedura *Update* din modulul *MirPers* a fost cu grijă evitată în exemplele pe care le-am prezentat. Este însă momentul să vă spun că această restricție este ridicată de o anumită facilitate la care vom ajunge în curând și că poate fi evitată de pe acum, folosind o variabilă auxiliară. Să completăm procedura *Test* din exemplul anterior (în locul marcat prin comentariul "etc") cu secvența:

```
p := a;
P.Update(p, "", "", "Boston");
WritePers(a)
```

Este simplu: am folosit o variabilă (*p*) având tipul static pretins de *Update* (adică *Pers*) dar având tipul dinamic *Ang*. Datorită faptului că *p* și *a* sunt pointeri, atribuirea *p := a* va face ca *p* să indice pe *a*, astfel încât procedura *Update* se va aplica de fapt acestuia din urmă. Rezultatul executării comenzii *MirParsTest.Test* va fi cel dorit:

```
John Icsulescu
varsta: 18
Adresa: Chicago

John Icsulescu
varsta: 18
Adresa: Boston
```

Merită să facem în acest moment o scurtă divagație. Vă puteți întreba, pe bună dreptate, de ce nu am definit tipul *Ang* într-un modul client al lui *MirPers*. Motivul este că, în acest caz, nu am fi putut – prin nici un subterfugiu – să inițializăm câmpul *born* (data nașterii), moștenit de la *Pers*, deoarece acesta nu este vizibil decât în *MirPers*. Această problemă (exprimată sub forma principiului "moștenirea sparge încapsularea") este larg dezbătută în literatura de specialitate. În Oberon, această problemă este rezolvată în conexiune cu o altă problemă generală ("fragilitatea clasei de bază") prin descurajarea moștenirii în favoarea compoziției. Tehnica pe care am folosit-o în exemple este bazată pe moștenirea implementării și este contraindicată în Oberon, dar este utilă pentru înțelegerea principiilor programării obiectuale. Voi reveni asupra acestei probleme la momentul oportun.

Gardienii tipurilor

Odată cu apariția tipurilor dinamice, întrevedem câteva posibilități foarte interesante de îmbogățire a exemplului dezvoltat până acum. De pildă, am dori ca procedura *WritePers* din modulul *MirPers1Test* să trateze diferit persoanele care sunt angajați. O îmbunătățire interesantă ar fi ca pentru acestea să se afișeze, în plus față de celelalte informații, și numărul de marcă (*id*). Pentru aceasta însă trebuie să putem determina din program tipul dinamic al variabilei *p* primite ca parametru. O variantă simplă o reprezintă utilizarea operatorului *IS* în cadrul unui ***test de tip***. Astfel, expresia *p IS Ang* va fi adevărată dacă tipul dinamic al variabilei p este *Ang* (sau eventuale extensii ale acestuia) și va fi falsă în caz contrar.

Pentru a rezolva și problema accesului la câmpul *id* al variabilei *p* (în cazul în care tipul dinamic al acesteia este *Ang*) avem nevoie de un ***gardian de tip*** (*type guard*). În cazul nostru acesta are forma *p(MirPers.Ang)*. Gardianul de tip are două roluri. În primul rând, verifică dacă tipul dinamic al variabilei este într-adevăr cel menționat în paranteză. Spre deosebire de testul de tip, gardianul de tip funcționează ca o ***aserțiune***, deci dacă tipul nu este cel așteptat va opri execuția programului. În al doilea rând, dacă tipul dinamic este corespunzător, gardianul de tip permite tratarea variabilei ca și cum tipul static ar fi același cu cel dinamic.

Procedura *WritePers* din modulul MirPersTest (vezi caseta "Aproape obiectual") poate fi deci completată cu instrucțiunea:

```
IF p IS P.Ang THEN
   Out.String("marca: "); Out.Int(p(P.Ang).id, 0); Out.Ln
END
```

unde *P* desemnează modulul *MirTest* (în care sunt definite tipurile *Pers* și *Ang*).

Este util să remarcăm că gardianul de tip este "protejat" de testul de tip din *IF*, astfel că niciodată acesta nu va avea posibilitatea să oprească programul. Să mai remarcăm că prin intermediul gardienilor de tip sunt posibile și atribuiri de genul *a := p(P.Ang)*, unde *a* este de tipul *Ang* iar *p* este de tipul *Pers* (dar având tipul dinamic *Ang*).

Mânuitorii de mesaje

Să ne aducem aminte că încă nu avem de-a face cu obiecte propriu-zise, deoarece încă nu am reușit să legăm explicit structura de date de procedurile care o procesează. Din fericire, avem deja la dispoziție toate instrumentele pentru a face acest pas (așa cum a fost el făcut în prima versiune a limbajului Oberon). Pentru aceasta, să renunțăm deocamdată la tipul *Ang* și să revenim la exemplul simplu din caseta "Aproape obiectual".

Cred că majoritatea cititorilor sunt familiarizați cu tipurile procedurale. Acestea nu există în standardul Pascal, dar sunt implementate în compilatoarele realizate de firma Borland (Turbo Pascal și Borland Pascal) exact în forma în care sunt prezente în Oberon.

Ideea este să dotăm variabilele de tip *Pers* cu o procedură care să primească din exterior comenzi prin intermediul unor mesaje și să acționeze în conformitate cu tipul și conținutul acestora. Pentru aceasta, vom modifica declarațiile de tip din modulul *MirPers* în felul următor:

```
TYPE
   Message* = EXTENSIBLE RECORD END;
   GetAgeMsg* = RECORD (Message) age*: INTEGER END;
   GetNameMsg* = RECORD (Message) name*:
                        ARRAY 80 OF CHAR END;
   UpdateMsg* = RECORD (Message)
      lname*, fname*, address*: ARRAY 80 OF CHAR
   END;

   Pers* = POINTER TO RECORD
      lname, fname, address-: ARRAY 80 OF CHAR;
      born: Dates.Date;
      handle*: PROCEDURE (p: Pers; VAR m: Message)
   END;
```

Interesant este că tipul *Message* nu are nici un câmp, servind doar ca bază de extensie pentru mesaje specifice. De asemenea, este de remarcat că mesajele sunt articole statice, spre deosebire de obiectele "active", care sunt implementate de regulă ca pointeri (explicația provine din faptul că variabilele statice sunt mai "rigide" în privința tipurilor dinamice: atribuirile între variabile statice nu afectează tipul dinamic).

După cum puteți bănui, rolul câmpului procedural *handle* este să găzduiască o procedură care să "mânuiască" mesajele (în engleză, "handler" înseamnă mânuitor, manevrant).

O implementare posibilă pentru un "handler" poate fi următoarea:

```
PROCEDURE HandlePers (p: Pers; VAR m: Message);
BEGIN
   WITH m: GetAgeMsg DO m.age := Age(p)
   | m: GetNameMsg DO GetName(p, m.name)
   | m:UpdateMsg DO
            Update(p, m.lname, m.fname, m.address)
   ELSE (* ignora mesajul *)
   END
END HandlePers;
```

Instrucțiunea *WITH* este (spre deosebire de Pascal) un test de tip combinat cu aplicarea implicită a unui *type guard* pe ramura respectivă. Deci, dacă tipul dinamic al lui *m* este, de exemplu, *GetAgeMsg*, atunci se execută instrucțiunile care urmează după *DO* ca și când tipul static al lui *m* ar fi *GetAgeMsg*. Separarea cazurilor se face prin simbolul "|" (bară verticală).

În caseta "Obiectual timpuriu" puteți urmări codul complet al modulului *MirTest* precum și codul unui mo-

Cu adevărat obiectual

```
MODULE MirPers;
    IMPORT Dates;
    TYPE
        Pers* = POINTER TO EXTENSIBLE RECORD
            lname, fname, address-: ARRAY 80 OF CHAR;
            born: Dates.Date
        END;
        Ang* = POINTER TO RECORD (Pers)
            id-, income: INTEGER
        END;

    PROCEDURE (p: Pers) GetName* (OUT n: ARRAY OF CHAR),
                                        NEW, EXTENSIBLE;
    BEGIN n := p.fname + ' ' + p.lname
    END GetName;

    PROCEDURE (a: Ang) GetName* (OUT n: ARRAY OF CHAR);
    BEGIN n := a.lname + ' ' + a.fname
    END GetName;

    PROCEDURE (p: Pers) Update* (IN lname, fname, address:
                                ARRAY OF CHAR), NEW;
    BEGIN
        IF lname # "" THEN p.lname := lname$ END;
        IF fname # "" THEN p.fname := fname$ END;
        IF address # "" THEN p.address := address$ END
    END Update;

    PROCEDURE (a: Ang) GetIncome* (): INTEGER, NEW;
    BEGIN RETURN a.income END GetIncome;

    PROCEDURE PInit* (IN lname, fname, address:
            ARRAY OF CHAR; born: Dates.Date): Pers;
        VAR p: Pers;
    BEGIN
        NEW(p); p.born := born;
        p.Update( lname, fname, address); RETURN p
    END PInit;

    PROCEDURE AInit* (IN lname, fname, address:
                ARRAY OF CHAR; IN born: Dates.Date;
                                id, income: INTEGER): Ang;
        VAR a: Ang;
    BEGIN
        NEW(a); a.born := born; a.id := id; a.income := income;
        a.Update( lname, fname, address); RETURN a
    END AInit;
END MirPers.

MODULE MirPersTest;
    IMPORT Dates, Out, P := MirPers1;

    PROCEDURE WritePers (p: P.Pers);
        VAR s: ARRAY 80 OF CHAR;
    BEGIN
        Out.Ln; p.GetName(s); Out.String(s); Out.Ln;
        Out.String("Adresa: " + p.address); Out.Ln;
        IF p IS P.Ang THEN
            Out.String("marca: "); Out.Int(p(P.Ang).id, 0); Out.Ln
        END
    END WritePers;

    PROCEDURE Test*;
        VAR a: P.Ang; p: P.Pers; d: Dates.Date;
    BEGIN
        d.year := 1980; d.month := 6; d.day := 21;
        p := P.PInit("Igrecescu", "Vasile", "Breaza", d); WritePers(p);
        a := P.AInit("Icsulescu", "John", "Chicago", d, 9154, 2500);
        WritePers(a)
    END Test;

END MirPers1Test.
```

dul client, *MirTest*, care testează funcționalitatea tipului *Pers*. Pentru a simplifica expunerea, am înlăturat câteva proceduri care n-au mai fost utilizate.

Observați că la inițializarea unui obiect (în funcția *Init*) se face și inițializarea câmpului procedural *handle* (exact ca în Turbo Pascal) și că nu mai este exportată nici o procedură, excepând funcția *Init* (necesară în exterior pentru a crea obiecte de tip *Pers*).

Urmărind acum modulul client, observăm că singura modalitate de a interacționa cu variabilele de tip *Pers* este prin intermediul *handler*-ului și al mesajelor. În felul acesta am reușit să conectăm explicit prelucrările de structura de date, astfel încât din acest moment putem spune răspicat că variabilele de tip *Pers* sunt obiecte.

Rezultatul executării comenzii *MirTest.Test* este cel așteptat. În fereastra *Log* este afișat textul

```
Ioana Icsulescu
Ioana Igrecescu
```

care probează corecta implementare a tipului *Pers*.

Ar mai fi câteva lucruri interesante de spus. În primul rând faptul că nu este nevoie ca mesajele să fie definite neapărat în modulul în care este definit tipul *Pers*. Orice modul care importă modulul *MirPers* poată să extindă tipul *MirPers.Message*, să instanțieze variabile din tipul nou creat și să le trimită obiectelor de tip *Pers*. Acesta nu le va recunoaște și, în consecință, le va ignora (observați ramura *ELSE* a instrucțiunii *WITH* din procedura *HandlePers*). Nu se va produce nici o eroare. Acest lucru este important atunci când un mesaj este difuzat într-o structură complexă de obiecte (procedeu numit *broadcasting*). Obiectele care nu-l înțeleg îl ignoră, iar cele care-l înțeleg răspund.

În fine, observăm că *handler*-ul este atribuit obiectului în mod dinamic (adică în timpul execuției). Nimic nu ne împiedică să schimbăm, tot în timpul execuției, *handler*-ul implicit al unui obiect cu un altul. Putem astfel să modificăm în mod dinamic comportamentul unui obiect. Oricât de tentant ar fi, procedeul este nerecomandabil, deoarece distruge omogenitatea în rândul obiectelor unei clase.

Voi încheia această secțiune cu câteva note privind terminologia. Teoria obiectuală spune că "obiectele comunică între ele prin intermediul mesajelor". Un mesaj definit teoretic ca un triplet de forma <*receptor*, *selector*, *argument*>, unde prin *receptor* se înțelege obiectul căruia îi este adresat mesajul, *selector*-ul desemnează procesarea care se cere executată, folosind (dacă este cazul) *argument*-ul. Nu se specifică nicăieri cum se implementează acest mecanism teoretic. În Oberon, implementarea s-a făcut respectând aproape textual această normă teoretică.

S-ar putea spune că mecanismul obiectual pe care l-am expus este sărăcăcios și primitiv. În schimb are avantajul simplității și al eficienței. Sistemul de operare Oberon cu care au fost echipate stațiile grafice Ceres a fost realizat

folosind în exclusivitate acest mecanism. Întregul cod sursă a avut circa 12.000 linii iar codul executabil arhivat încape pe o dischetă.

Proceduri tipizate

Principala noutate pe care a adus-o Oberon-2 (alături de tablourile deschise) a constituit-o introducerea procedurilor "legate de tip" (sau "atașate unui tip" - *type-bound procedures*), pe care eu le voi numi în continuare proceduri tipizate. Acestea conduc la o explicitare mai accentuată a legăturii dintre structura de date și procesările care-i sunt specifice. În limbaj obiectual, procedurile tipizate sunt numite ***metode***.

Pentru exemplificare, vom renunța la *handler* și vom reveni la o versiune mai veche a tipului *Pers* :

```
TYPE
   Pers* = POINTER TO EXTENSIBLE RECORD
      lname, fname, address-: ARRAY OF CHAR;
      born: Dates.Date
   END;
```

Procedura *GetName*, în varianta tipizată, va arăta astfel:

```
PROCEDURE (p: Pers) GetName* (OUT n: ARRAY OF CHAR),
                                          NEW;
BEGIN
   n := p.fname + ' ' + p.lname
END GetName;
```

Așadar, tipul căreia îi este atașată procedura se scrie înaintea numelui procedurii. Este important de remarcat că tipul se specifică prin intermediul unui parametru formal (în acest caz *p*) astfel încât în interiorul procedurii tipizate ne vom referi la obiectul afectat prin acesta. În alte limbaje obiectuale referirea se face printr-un identificator generic (de pildă *Self*).

Marcajul *NEW* este o extensie Component Pascal prin care se desemnează metodele nou introduse, pentru a fi ușor distinse de cele redefinite (vezi mai jos).

În rest, lucrurile sunt simple. Apelarea (în limbaj obiectual, ***invocarea***) procedurii pentru a fi aplicate unui obiect *p* de tip *Pers* se face astfel:

```
p.GetName(s);
```

Exprimarea este simplă și foarte sugestivă, deoarece pune în evidență faptul că procedura aparține efectiv obiectului *p*, la fel ca un câmp. Pe de altă parte, merită să menționez că un astfel de apel este, de fapt, ceea ce teoria OO numește "mesaj". În acest caz, *p* este receptorul, *GetName* este selectorul iar *s* este argumentul.

Faptul că metodele aparțin obiectului la fel ca și câmpurile nu este doar formal, ci are o consecință de o importanță crucială: extensiile unui tip (în cazul nostru *Pers*) vor moșteni nu doar câmpurile, ci și metodele tipului de bază. În acest sens lărgit, ***moștenirea*** (*inheritance*) este un concept fundamental al tehnologiei obiectuale, reprezentând fundamentul pentru unul dintre scopurile principale ale acesteia: reutilizarea.

Moștenirea

În caseta "Cu adevărat obiectual" am încercat să pun în evidență câteva aspecte ale moștenirii. Modulul *MirPers* cuprinde definiția tipului *Pers* utilizând proceduri tipizate (din considerente de spațiu am renunțat la unele metode), împreună cu definiția unui descendent al acestuia, tipul *Ang*.

Din faptul că *Ang* este o extensie a tipului *Pers* (în limbaj obiectual; *Ang* este un descendent – sau "o subclasă" – a clasei *Pers*) rezultă că va dispune de toate metodele acestuia. Astfel, procedura tipizată *Update* va fi aplicabilă oricărei instanțe a clasei *Ang*.

Este însă foarte probabil ca o extensie a unui tip, fiind în general o specializare a acestuia, să aibă nevoie și de metode proprii, care se adaugă celor moștenite. În cazul nostru, tipul *Ang* dispune de metoda *GetIncome()*, care nu are sens pentru un obiect având tipul dinamic *Pers*.

În fine, este de asemenea posibil ca o subclasă să nu fie satisfăcută de unele dintre clasele pe care le moștenește. În cazul acesta poate să le redefinească (uneori se folosește termenul "supraîncărcare" – *overloading*). În exemplul din caseta "Cu adevărat obiectual" am presupus că ar fi mai potrivit ca numele angajaților să fie extras în forma <nume + prenume > (și nu invers, cum a fost definit pentru persoane în general). Din acest motiv, am redefinit metoda *GetName*. Astfel, obiectelor având tipul dinamic *Ang* li se va aplica metoda redefinită, în timp ce obiectelor având tipul dinamic *Pers* li se va aplica în continuare metoda originală. Dacă dorim totuși ca într-o metodă pe care o redefinim să apelăm metoda clasei părinte, o putem accesa în forma *p.GetName^ (s)* (acest procedeu se numește *supercall* și în Oberon este permis, dar nerecomandat).

Notați cu acest prilej câteva aspecte ale sintaxei Component Pascal (e vorba de extensii care nu aparțin standardului Oberon-2). Definiția metodei originale (cea definită pentru *Pers*) este marcată ca *NEW* și *EXTENSIBLE*. Rolul acestor marcaje este oarecum formal, dar eficient. Marcajul (atributul) *NEW* trebuie folosit atunci când intenționam să adăugăm o metodă nouă. Deoarece o clasă poate moșteni multe metode, de la mai mulți strămoși (în limbaj obiectual: ancestori) este posibil să alegem – din neștiință sau din greșeală – un nume care coincide cu o metodă deja existentă. În felul acesta vom redefini involuntar metoda, ceea ce poate conduce la erori ascunse foarte greu de depistat (și care se vor propaga prin moștenire). Atributele *EXTENSIBLE* și *NEW* ne obligă să luăm în mod conștient decizii clare privind arhitectura viitoare a aplicației.

Pur sau hibrid ?

Oberon este un limbaj hibrid. Acest lucru poate fi evidențiat printr-o comparație cu cerințele minimale pe care, teoretic, ar trebui să le îndeplinească în limbaj obiectual "pur". Aceste cerințe pot fi exprimate prin următoarele axiome:

Axioma 1:
a) Orice este obiect. Un obiect se poate caracteriza prin:
b) o structură de date, adică un set de variabile care definesc starea obiectului și sunt rezervate pentru a fi folosite în mod privat de către acesta.
c) o structură de control al comportamentului, adică un set de mesaje, care sunt în general publice. Ele pot fi folosite de către alte obiecte pentru a comunica cu obiectul în cauză. Atunci când sunt primite de obiect, mesajele activează metodele care le sunt asociate, care determină un anumit comportament al obiectului.
Axioma 2:
a) Orice obiect este creat pornind de la un model care-i este propriu și care se numește clasă. Se spune despre un obiect că este o instanță a clasei sale.
b) Unele obiecte au statut dublu, fiind ele însele clase. Clasele care le corespund se numesc metaclase.
Axioma 3: Orice obiect acceptă mesaje (cu sau fără parametri) la care răspunde.
Axioma 4: Orice clasă este o subclasă a (cel puțin) unei alte clase (superclasa), exceptând clasa "Obiect", care este rădăcina ierarhiei

Oberon nu îndeplinește axioma 1.a., deoarece doar instanțele unui tip RECORD sunt considerate obiecte. În Smalltalk, de pildă, chiar și o constantă întreagă este un obiect.
Axioma 1.b. poate fi considerată ca fiind satisfăcută de Oberon, cu observația că proiectantul poate să ofere acces public la starea obiectelor (prin export).
Axioma 1.c. este îndeplinită integral, indiferent dacă este vorba de obiecte cu metode sau obiecte cu handler.
Axioma 2.a. este îndeplinită integral, în timp ce axioma 2.b. nu este îndeplinită deloc.
Îndeplinirea axiomei 3 depinde de interpretare. Dacă vom considera "obiect" doar instanțe ale tipurilor RECORD înzestrate cu cel puțin o metodă (sau cu un handler) atunci o putem considera satisfăcută. Dacă vom considera "obiect" orice instanță a oricărui tip RECORD, atunci articolele "banale" (fără metode) nu pot primi mesaje, deci axioma nu este satisfăcută.
Axioma 4 nu este satisfăcută. În Oberon, clasele pot forma ierarhii multiple, pornind de la oricâte rădăcini. Component Pascal a introdus tipurile implicite ANYREC și ANYPTR, care sunt considerate în mod implicit tip de bază al oricărui tip RECORD sau POINTER TO RECORD. Aceste tipuri sunt utile doar pentru operații speciale de inginerie software (refactorizarea unui întreg framework sau subsistem). Să mai notăm că Oberon nu respectă nici normele privind modularitatea la nivel de clasă (un modul Oberon poate cuprinde mai multe clase) și nu suportă moștenirea multiplă, care este compensată prin posibilitatea de compunere.

Abstractizare

Este important de subliniat că mecanismul moștenirii ne permite (și chiar ne încurajează) să factorizăm elementele comune ale unei ierarhii de clase într-o clasă generică, cu caracter general. Un exemplu simplu va ridica repede ceața. Să presupunem că proiectăm o aplicație care lucrează cu figuri geometrice plane și că pentru acestea ne interesează anumite elemente cum ar fi aria, perimetrul etc. Dreptunghiul și cercul (de exemplu) vor avea metode diferite pentru calculul acestor elemente, dar ne interesează ca ele să fie compatibile. Putem să începem prin a defini o clasă numită *Figura*, care va avea o poziție (dată, să zicem de coordonatele *x* și *y* ale centrului de greutate). Știm că fiecare figură particulară va avea metode specifice pentru calculul ariei și al perimetrului, dar pentru o figură generică nu putem să precizăm aceste metode. Avem însă posibilitatea să definim procedurile *Aria* și *Perimetru* ca ***metode abstracte***. Ele nu vor avea o implementare pentru clasa *Figura*, dar vor fi moștenite și implementate de fiecare descendent al acesteia (de pildă *Dreptunghi*, *Cerc* etc.).

O clasă care are cel puțin o metodă abstractă este o ***clasă abstractă*** (și în Component Pascal trebuie marcată ca atare, cu atributul *ABSTRACT*). Deducem că o clasă abstractă poate avea și metode care nu sunt abstracte (sunt concrete). Putem, de exemplu, să definim pentru clasa *Figura* o metodă (concretă) pentru schimbarea poziției. Aceasta poate fi implementată deoarece nu implică decât coordonatele plane (care sunt definite în *Figura*) și poate fi moștenită de toate subclasele acesteia. În Component Pascal putem chiar să interzicem redefinirea acesteia în clasele descendente (nu o declarăm *EXTENSIBLE*).

De notat că o clasă abstractă nu poate fi instanțiată iar invocarea unei metode abstracte provoacă o eroare de execuție (fatală!). Oarecum asemănătoare cu metodele abstracte sunt metodele vide (marcate în Component Pascal cu atributul *EMPTY*). Nici acestea nu au o implementare dar, spre deosebire de metodele abstracte, pot fi invocate, însă fără nici un efect. O metodă de genul *Roteste* va putea fi, în cazul cercului, o metodă *EMPTY*.

Polimorfism

Revenind la exemplul cu persoane și angajați, să remarcăm ca putem trata în mod uniform un șir de persoane (angajați sau nu), obținând rezultate potrivite fiecărui caz particular. De pildă, aplicând tuturor persoanelor metoda *GetName* vom obține numele complet al persoanei fie ca <prenume+nume> fie ca <nume+prenume>. Acest comportament uniform dar diferențiat se numește ***polimorfism*** și este încă una dintre trăsăturile programării obiectuale (fără a fi specifică).

Nu ar fi lipsit de interes să ne oprim o clipă asupra modului în care este implementat polimorfismul în Oberon. Pentru aceasta trebuie mai întâi să clarificăm un aspect: Procedurile tipizate (metodele) aparțin de fapt

tipului (clasei) și nu obiectelor (instanțe ale clasei). Toate instanțele clasei au exact aceleași metode, implementate la fel, așa că nu este nevoie ca fiecare obiect să poarte referințe la metodele sale (cu atât mai puțin implementări). Remarcați că situația nu este aceeași pentru obiectele implementate cu ajutorul *handler*-elor, unde instanțe ale aceluiași tip puteau avea *handler*-e diferite.

Pentru variabilele care nu pot avea un tip dinamic diferit de tipul static (deoarece aparțin unui tip care nu are descendenți), compilatorul poate stabili cu certitudine care sunt metodele care vot fi activate. Această modalitate de a lega variabila de metodele care-i vor fi aplicate se numește ***legare statică*** și este comună tuturor limbajelor. În cazul limbajelor obiectuale, apare situația în care o aceeași variabilă poate conține obiecte de mai multe tipuri, adică tipul static al variabilei (cel cu care a fost declarată în program) nu coincide cu tipul dinamic (din momentul execuției), deci compilatorul nu poate stabili întotdeauna o corespondență între variabilă și metodele care-i vor fi aplicate. Mai precis, problema se pune în cazul metodelor adăugate sau redefinite în cadrul extensiilor tipului static, deoarece pentru celelalte metode este limpede că se aplică metoda definită în tipul de bază. În celelalte cazuri însă, stabilirea metodei care trebuie invocată nu poate fi făcută decât la momentul execuției. Procedeul prin care se realizează aceasta se numește ***legare dinamică*** (sau "târzie" – în engleză *late binding*). Implementarea acestui mecanism diferă de la limbaj la limbaj și de la compilator la compilator, dar se bazează pe faptul că, într-un fel sau altul, un obiect poartă cu sine niște informații privind tipul căruia îi aparține. În Oberon, variabilele dinamice (pointerii) indică o structură de date care, pe lângă câmpurile articolului, conțin și un pointer (inaccesibil programatorului) la o altă structură de date, numită ***descriptor de tip*** (unic pentru fiecare tip). Aceasta, la rândul ei, cuprinde pointeri la metodele (proprii) tipului precum și la descriptorii tipurilor descendente (important: conține și numele tipului). La invocarea unei metode, o rutină specială a mediului de execuție va găsi descriptorul de tip (pe baza pointerului din cadrul obiectului) și va căuta metoda invocată. Dacă o va găsi, o va lansa în execuție. Dacă nu, va deduce că este o metodă moștenită și va căuta superclasa imediat superioară, unde va repeta operația.

Atenție! o variabilă dinamică ce nu a fost încă alocată (prin NEW) va avea valoarea NIL, deci nu va puncta la nici o structură de date. Rezultă că nu va dispune nici de pointerul la descriptorul de tip, deci nu îi poate fi aplicată nici o metodă. O consecință este faptul că în Oberon nu există ***constructori*** de genul color din C++ sau Java.

Interfețe și implementări

Am menționat deja că în Oberon este contraindicată moștenirea implementării, fiind în schimb practicată moștenirea interfeței. Aceasta se traduce prin faptul că în Oberon este contraindicată exportarea caselor concrete. Există însă două excepții, dintre care una am întâlnit-o deja: mesajele sunt exportate întotdeauna ca tipuri concrete (explicația este faptul că, prin natura lor, mesajele nu au un comportament propriu).

Toate exemplele de până acum s-au bazat pe exportarea unor tipuri concrete (și totuși funcționează). Problema este însă că în acest fel, modulele care utilizează tipurile pe care le-am implementat sunt dependente de această implementare, iar în situația în care această implementare se schimbă este posibil ca modulele client să fie afectate. O rezolvare mai elegantă a problemei este ilustrată în continuare:

```
TYPE
   Pers* = POINTER TO ABSTRACT RECORD
      lname, fname, address-: ARRAY OF CHAR;
      born: Dates.Date
   END;

   StdPers = POINTER TO RECORD (Pers) END;
```

Am transformat deci tipul *Pers* într-unul abstract și am definit un tip concret, *StdPers*, care-l extinde. Acest tip nu va fi exportat și va servi doar pentru implementare. Toate metodele tipului *Pers* le declarăm abstracte și mutăm implementarea în metodele corespunzătoare ale tipului concret *StdPers*. De exemplu, metodele *GetName* vor fi declarate astfel:

```
PROCEDURE (p: Pers) GetName* (OUT n: ARRAY OF CHAR),
                                   NEW, ABSTRACT;

PROCEDURE (p: StdPers) GetName (OUT n: ARRAY OF CHAR);
BEGIN n := p.fname + ' ' + p.lname
END GetName;
```

Vă întrebați de bună seamă cum conectăm cele două tipuri, astfel încât invocarea unei metode a tipului *Pers* să fie redirectată tipului *StdPers*. Aceasta se realizează prin funcția *Init*, care exportată pentru a permite crearea obiectelor de tip *Pers*:

```
PROCEDURE Init* (IN lname, fname, address:
      ARRAY OF CHAR; born: Dates.Date): Pers;
   VAR p: StdPers;
BEGIN
   NEW(p); p.born := born;
   p.Update( lname, fname, address); RETURN p
END Init;
```

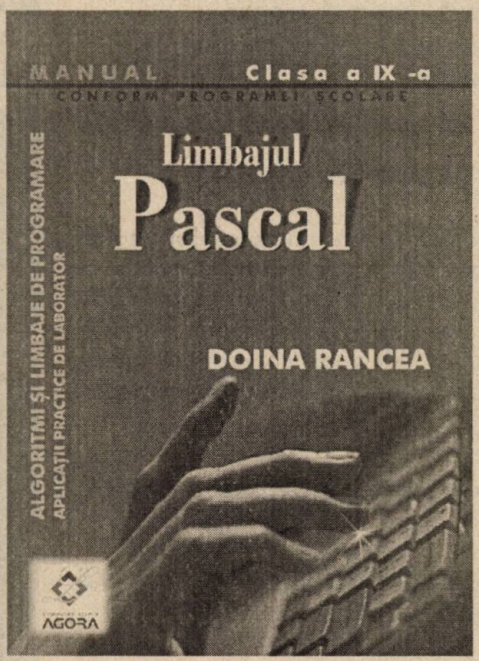
Elegant, nu-i așa?

Mircea Sârbu este directorul editorial al editurii Computer Press Agora.
E-mail: msarbu@agora.ro.

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Limbajul Pascal este un manual care acoperă programa școlară (1997) a două discipline: „Algoritmi și limbaje de programare" și „Aplicații practice de laborator" care se studiază în clasele a IX-a cu profil de informatică.

Manualul prezintă noțiunile din programă la un nivel accesibil, fiecare noțiune fiind ilustrată în mai multe programe Borland Pascal 7.0. Aceste programe sunt însoțite și de prezentarea datelor de intrare și a rezultatelor în forma sub care apar pe ecran. Problemele au fost selectate în așa fel încât să ilustreze multiplele domenii de utilizare a calculatoarelor. Nu lipsesc problemele de simulare, jocuri, probleme distractive, dar au fost alese și aplicații din programa de matematică de clasa a IX-a cu scopul exemplificării și fixării unor noțiuni importante din matematică. În manual se insistă asupra felului în care se descoperă rezolvarea unei probleme, nivelul de dificultate a acestora crescând gradat, venind astfel în sprijinul elevilor și profesorilor.
Numeroasele probleme din manual permit utilizarea acestuia nu numai pentru studiu individual ci și ca bogat material didactic pentru lecțiile de programare și laborator.
Manualul se adresează elevilor, studenților, profesorilor și oricărei persoane interesate de programare.
Manualul a fost conceput, plecând de la ediția din 1993 (Limbajul Turbo Pascal, clasa a X-a) care a fost regândit substanțial și îmbogățit, astfel încât să fie accesibil unui elev de clasa a IX-a. Ediția anterioară s-a bucurat de un succes deosebit, fiind apreciată de mii de cititori.

Autoarea, Doina Rancea, este absolventă cu diplomă de merit a Facultății de Matematică, Secția Informatică a Universității Babeș-Bolyai din Cluj Napoca (1975). A fost repartizată la Liceul de Informatică din Cluj Napoca unde predă și în prezent limbajele de programare Pascal și C++. Este profesoară gradul I și în această calitate îndrumă activitatea de practică pedagogică a studenților. Experiența bogată a autoarei a fost valorificată cu ocazia participării, în cadrul Comisiei Naționale de Specialitate Informatică a Ministerului Educației Naționale, la actualizări ale programei școlare pentru disciplina informatică și în cadrul comisiilor județene și naționale ale olimpiadelor, propunând probleme originale.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Vă place GInfo?

Nume: _____ Prenume: _____ Vârsta: _____

Adresa: _____

Localitate: _____ Cod poștal: _____ Județ: _____ Telefon: _____

Ocupația: elev informatică ☐ elev ☐ student ☐ cadru didactic ☐ altă ocupație ☐ (precizați: _____)

Școala (firma, organizația etc.): _____

Dotări

Folosiți un calculator acasă? Da ☐ Nu ☐

Câte calculatoare sunt instalate în școală: (aprox.) _____

Aveți acces la e-mail: Da ☐ Nu ☐

Aveți acces la WWW: Da ☐ Nu ☐

Cum apreciați dotarea cu tehnică de calcul a școlii: _____

Aprecieri generale

Acordați note de la 1 la 10 pe următoarele criterii:

Structura tematică: ___

Calitatea articolelor: ___

Prezentare grafică: ___

Calitate tipografică: ___

Considerați că prețul este:

prea mare ☐ accesibil ☐ prea mic ☐

De cât timp cunoașteți Gazeta de Informatică (ani)? _____

Cum ați procurat acest număr:

chioșc ☐ abonament ☐ cadou ☐ altfel ☐

În acest număr

Apreciați cu note de la 1 la 10 articolele acestui număr:

Arbori Steiner ___

Problema fluxului maxim ___

Object Pascal și Delphi ___

Indexare cu "hăți de biți" ___

Motoare de căutare ___

Serial Java ___

Pagini WEB cu HTML 3.2 ___

Oberon: obiecte și tipuri ___

Articolul care v-a plăcut cel mai mult: _____

De ce? _____

Articolul care v-a plăcut cel mai puțin: _____

De ce? _____

Considerați că ponderea problemelor rezolvate este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea problemelor propuse este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea subiectelor teoretice este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că nivelul de accesibilitate al articolelor este

prea cobotât ☐ potrivit ☐ prea ridicat ☐

În viitor

În următoarele numere așteptați:

mai multe articole de programare? Da ☐ Nu ☐

mai multe articole teoretice? Da ☐ Nu ☐

mai multe articole despre Internet? Da ☐ Nu ☐

mai multe prezentări de programe? Da ☐ Nu ☐

prezentări de jocuri? Da ☐ Nu ☐

articole despre hardware? Da ☐ Nu ☐

V-ar plăcea să citiți:

pagini redactate de elevii unor licee? Da ☐ Nu ☐

lucrări premiate, realizate de elevi? Da ☐ Nu ☐

actualități școlare? Da ☐ Nu ☐

prezentarea unor situri Web? Da ☐ Nu ☐

tips & tricks? Da ☐ Nu ☐

Comentarii:

_____

_____

_____

_____

_____

_____

_____

Dacă nu vă este suficient spațiul, folosiți coli albe suplimentare.

Poșta redacției

Cu speranța că pe parcurs, conturile de e-mail de prin școli se vor mai stabiliza, am redeschis lista de discuții *Dialog*. Vă reamintesc că pentru a vă abona nu trebuie decât să trimiteți un mesaj e-mail la adresa **majordomo@netsoft.ro**. În corpul mesajului scrieți "subscribe dialog". Vă așteptăm.

Adevărul este că, peste vară, corespondența sosită la adresa redacției a scăzut simțitor. Deschiderea listei de discuții *Dialog* a fost amânată, deoarece testele din luna septembrie au indicat un număr foarte mare de mesaje de eroare ("adresă inexistentă"). Dacă ești abonat și până la momentul în care citești aceste rânduri nu ai primit un mesaj cu titlul "Welcome back", înseamnă că ceva nu e OK și ar fi bine să te abonezi din nou.

Dorința noastră este ca, spre deosebire de anul școlar trecut, temele de discuție să fie propuse mai ales de către voi, cititorii. Nu ezitați să vă spuneți deschis părerea despre revistă și mai ales, să ne sugerați modalități de a o face mai interesantă. La fel ca anul trecut, vom oferi mici cadouri (CD-uri sau cărți) celor care ne-au reținut în mod special atenția prin scrisorile pe care ni le-au trimis.

Completare la interviuri

În numărul precedent al GInfo vă invitam să adresați și voi întrebări invitaților noștri de la rubrica *Interviu*. În urma acestei invitații, cititorul și colaboratorul nostru *Tiberiu Socaciu* din Cluj-Napoca a adresat câte o întrebare suplimentară d-nei Irina Athanasiu și d-rei Anca Voicu. Deoarece răspunsurile au venit foarte repede (pentru care redacția mulțumește celor două doamne), putem face în această pagină completările de rigoare.

Întrebare adresată d-nei *Irina Athanasiu*: *"Cunosc poziția dvs. relativ la calitatea învățământului, dascălilor și studenților. În aceste ipoteze (pe care le-ați făcut publice, de altfel), dacă studentul X - pe care-l "vânați" pentru un post de preparator la catedra dvs. - vă cere sfatul în anul terminal: «D-nă profesor, am primit o ofertă de lucru în țara Y, unde voi fi foarte bine plătit, ce să fac?»; ce îi răspundeți?"*

Răspuns: "Situația pe care o descrieți nu este de loc atât de ipotetică cum credeți. Mi se întâmplă în fiecare an, și încă de mai multe ori. Ceea ce fac de fiecare dată este să încerc să aduc argumente *pro* și *contra*. Un singur lucru este clar. Fiecare are dreptul să își aleagă drumul în viață care să îi permită o dezvoltare cât mai bună, pe plan profesional, personal și uman, în general. Generația mea a auzit mereu că este o generație de sacrificiu. Nimeni nu ne-a întrebat dacă aceasta este ceea ce ne dorim. Nu cred că avem dreptul să le facem același lucru celor mai tineri decât noi. [...]"

Întrebare adresată d-rei *Anca Voicu*: *"Dumneavoastră, ca absolventă de inginerie care ați reușit să intrați în învățământul preuniversitar, cum priviți refuzarea acceptării absolvenților de calculatoare la concursurile de titularizare pe catedre de informatică? Este cazul unor colegi de-ai mei de generație, care aveau în urmă, totuși, câțiva ani de catedră."*

Răspuns: "Întrebarea *«Cine să predea informatica?»* se pune de câțiva ani. Au mai fost ani în care nu s-a permis inginerilor automatiști să participe la concurs. Și totuși, de câțiva ani (mai precis, de când sunt eu în Minister) acest lucru a fost posibil. Din păcate, noul Statut al Cadrelor Didactice impune ca posturile să fie ocupate în funcție de calificarea obținută. Învățământul superior a stabilit printr-o hotărâre de guvern cine poate preda fiecare disciplină în parte, fără a ne consulta.

Pentru anul viitor se fac demersuri pentru a reglementa această problemă."

O restanță

Într-un număr precedent al revistei (GInfo 4-5/98) l-am premiat pe câștigătorul premiului de 200.000 lei pentru cea mai bună implementare a unui *unit* pentru data calendaristică. Din păcate, s-au strecurat câteva greșeli. Numele său este *Adrian Sandor* (și nu Șandor) și este student la Universitatea din București (și nu la Politehnică). Ne cerem scuze. Tot Adrian ne semnalează o serie de *bug*-uri din numărul precedent. O nouă organizare redacțională care a intrat în vigoare odată cu acest număr va reduce cu siguranță numărul acestor greșeli.

Distribuția

Numeroși cititori s-au arătat nemulțumiți de modul în care revista este distribuită în anumite zone ale țării. Trebuie să facem precizarea că editura nu controlează decât distribuirea unei părți a tirajului, o altă parte fiind difuzată prin rețele independente de distribuție a presei. Din acest motiv nu putem promite o îmbunătățire imediată a situației, deși am dori ca acest lucru să se întâmple.

Reamintim cititorilor că modalitatea cea mai sigură și cea mai avantajoasă de a procura revista este **abonamentul**.

Mircea Sârbu

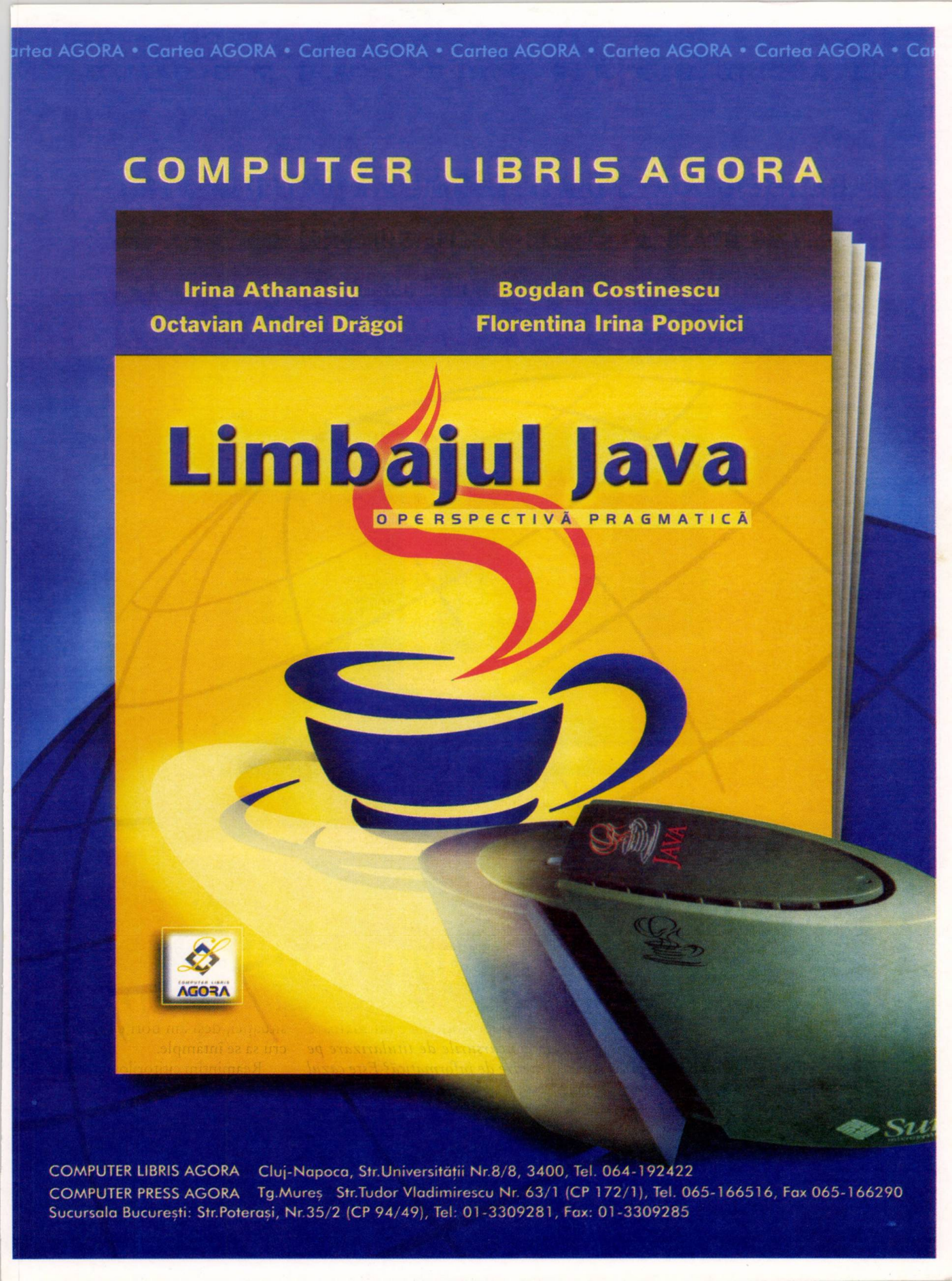
artea AGORA • Cartea AGORA • Cartea AGORA • Cartea AGORA • Cartea AGORA • Cartea AGORA • Car

COMPUTER LIBRIS AGORA

Irina Athanasiu
Octavian Andrei Drăgoi
Bogdan Costinescu
Florentina Irina Popovici

Limbajul Java

O PERSPECTIVĂ PRAGMATICĂ



COMPUTER LIBRIS AGORA Cluj-Napoca, Str.Universității Nr.8/8, 3400, Tel. 064-192422
COMPUTER PRESS AGORA Tg.Mureș Str.Tudor Vladimirescu Nr. 63/1 (CP 172/1), Tel. 065-166516, Fax 065-166290
Sucursala București: Str.Poterași, Nr.35/2 (CP 94/49), Tel: 01-3309281, Fax: 01-3309285

Ştiţi cât costă un IBM PC?

doar 1100 USD !!!

Dacă pe lângă cei **800** USD cât costă un pachet **LOTUS SmartSuite** mai plătiţi **300** USD, primiţi şi un calculator original **IBM PC300GL**, în următoarea configuraţie:

- procesor INTEL Pentium MMX 200MHz
- 16MB RAM
- 2,1GB HDD
- placă video PCI 1MB VRAM
- monitor IBM G42 14"
- tastatură IBM 104 taste UK
- mouse IBM
- Windows 95 preinstalat, cu licenţă
- licenţă LOTUS SmartSuite

şi nu este singura noastră ofertă !!!

IBM Business Partner

Microsoft
WHERE DO YOU WANT TO GO TODAY?™
Authorized Distribution Partner/
DSP for Romania

FORTE COMPUTERS
Flexibility.Made in Romania

IBM Business Partner

Lipscani 102, Bucureşti
Tel: 3122360/6137282/3120948 Fax: 3122630

Gifo

Noiembrie 1998
Vol. 8/8

Lei 10000

Gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

DIGRAFURI caracterizate prin K-formule

Înfășurătoarea convexă în 2D

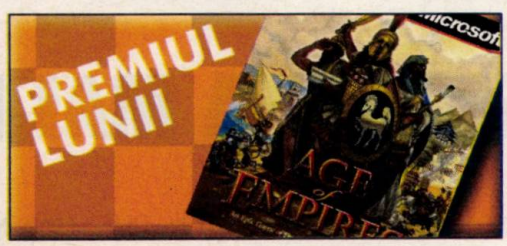
Programarea PLĂCII DE SUNET

TEHNO
Macrouri în WORD'97

ACTUAL
TI&C
în reforma curriculară

PROBLEME
CEOI'98
Zagreb - Croația

INTERVIU
PRIVIRE ÎNAPOI
Mihai SCORȚARU



T. P.
Aprobat R.A.P.R.
Nr. 103/DC/053/1998
Valabil 1998

Expeditor
s.c. „Computer Press Agora" SRL
C. P. 172 O.P. 1
RO-4300 Tirgu-Mureș

INFADRES

COMPUTER PRESS AGORA

ÎNTR-O LUME ÎN PERPETUĂ SCHIMBARE,

ÎN CARE LUNAR SUNT LANSATE PE PIAȚĂ

MII DE TEHNOLOGII ȘI PRODUSE NOI,

AI NEVOIE DE O **CĂLĂUZĂ!**

PC MAGAZINE ROMÂNIA

te ajută oferindu-ți tot ceea ce-ți este necesar pentru a te orienta printre produsele și tehnologiile informatice de ultimă oră. PC Magazine România îți oferă articole de prezentare și evaluare; teste de laborator și analize comparative, opinii ale unor experți recunoscuți și îți prezintă într-o manieră accesibilă cele mai moderne tehnologii.

Citește **PC MAGAZINE ROMÂNIA** și vei găsi

CALEA SPRE CEA MAI BUNĂ ALEGERE

COMPUTER PRESS AGORA, Str.Tudor Vladimirescu Nr.63/1, 4300 Tg.Mureș, Tel. 065-166516, Fax 065-1662190

Vol. 8, Nr. 8 1998

gazeta de informatică

Redacția:

Director științific:
Prof. dr. *Horia Georgescu* (București)
Redactor șef:
prof. *Clara Ionescu* (Cluj)
Editor executiv:
Mircea Sârbu (Tg-Mureș)
Au mai colaborat:
Ion Cozac (Târgu-Mureș)
Anca Voicu (București)
Mihai Scorțaru (Cluj-Napoca)
Adrian Monea (Cluj-Napoca)
Moise Cocan (Brașov)
Anca Vasilescu (Brașov)
Mihai Oltean (Cluj-Napoca)

Tehnoredactare:
Dan Crețu, Călin Borz
Coperta și grafica:
Mircea Drăgoi

Adresa pentru corespondență:
C.P. 172-1, 4300 Tg-Mureș
Adresa e-mail: office@ginfo.ro
Tel: (065) 166.516
Fax: (065) 166.290
http://www.ginfo.ro

ISSN: 1222-7129
Tiparul: *S.C. Infopress srl*,
Odorheiu Secuiesc,
tel: 066.216.483
Editor: Computer Press Agora srl
str. T. Vladimirescu nr. 63/1
4300 Târgu-Mureș.
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director editorial:
Mircea Sârbu
Director tehnic:
Szabó László
Vânzare reclamă:
Mihaela Szabo (mszabo@agora.ro)
Denis Marioara (mdenis@agora.ro)
Reprezentanța București:
Director: *Dana Pastramă*
Tel: (01) 330.92.81

Σ Sumar

soluții.. PAG. 4

probleme REZOLVATE
PAGINI REALIZATE DE
Clara IONESCU
Algoritmi și implementări propuse de autorii problemelor

probleme.. PAG. 17

propuse pentru REZOLVARE
***
Problemele de la Olimpiada de Informatică a Europei Centrale '98
Zadar - Croația

interviu.. PAG. 21

O scurtă privire înapoi
De la școală la facultate
Interviu cu *Mihai SCORȚARU*

actual.. PAG. 24

TI & C

în reforma curriculară
Anca VOICU

focus.. PAG. 27

Digrafuri caracterizate prin K-formule.
Aplicație în Delphi
Moise COCAN și Anca VASILESCU

Înfășurătoarea convexă în 2D
Ion COZAC

tehno.. PAG. 38

Macrouri în Word '97
Boldarea cuvintelor cheie în textele sursă tipărite
Mihai OLTEAN

Programarea plăcii de sunet
Adrian MONEA

serial.. PAG. 47

Lucrul cu șiruri în JAVA
Horia GEORGESCU

Pagini WEB cu HTML 3.2
Mihai SCORȚARU

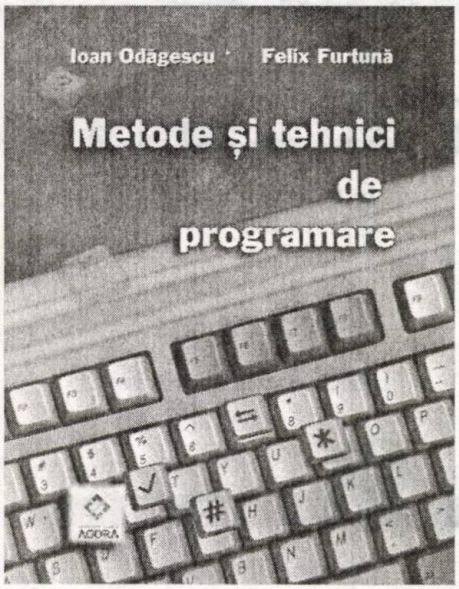
babel.. PAG. 57

Oberon:
Structuri generice
Mircea SÂRBU
Mai mult decât un limbaj: o filosofie.

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Prof. univ. dr. ***IOAN ODĂGESCU***, este absolvent al *Facultăților de Cibernetică, Statistică și Informatică Economică* (***Academia de Studii Economice***) și *Matematică-Mecanică* (***Universitatea București***).

Cu o vechime de peste 25 de ani în învățământul superior de Informatică Economică, membru al Catedrei de Informatică Economică din ASE, a publicat multe lucrări de referință pe diferite domenii, cum ar fi:

În domeniul analizei numerice:
METODE NUMERICE ȘI SUBRUTINE
OPTIMIZAREA ALGORITMILOR

În domeniul informaticii:
VIRUȘI INFORMATICE ȘI BOMBE LOGICE – prima lucrare de acest gen din țară
LIMBAJUL COBOL FĂRĂ PROFESOR
SIMULAREA PROCESELOR ECONOMICE, GRAFICA INTERACTIVĂ PE CALCULATOARE PERSONALE
METODE ȘI TEHNICI DE PROGRAMARE – enunțuri, soluții, probleme propuse <<cartea galbenă>>)

Trebuie remarcat faptul că, domnul profesor a coordonat colectivul care a publicat prima lucrare pe plan național, întitulată OLIMPIADE NAȚIONALE ȘI INTERNAȚIONALE DE INFORMATICĂ

Domnul prof. Ioan Odăgescu este conducător de doctorat în specialitatea Informatică Economică. În cadrul acțiunilor științifice de promovare a tinerelor talente, de aproximativ 20 de ani se ocupă cu organizarea concursurilor de informatică și este autor a multor probleme date la concurs.

Este membru în consiliul ***Societății Române de Informatică*** (SOCROM INFO) și membru ***ACM*** (USA).

* * *

Asist. univ. drd. ***FELIX FURTUNĂ*** este absolvent al *Facultății de Cibernetică, Statistică și Informatică Economică* (***Academia de Studii Economice***, București), specializarea *Informatică Economică*. În prezent este cadru didactic la *Facultatea de Cibernetică Statistică și Informmatică Economică*, catedra de Informatică Economică.

Este coautor al lucrărilor:
METODE ȘI TEHNICI DE PROGRAMARE; OLIMPIADE NAȚIONALE ȘI INTERNAȚIONALE DE INFORMATICĂ
GRAFICA INTERACTIVĂ PE CALCULATOARE PERSONALE

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422] sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

SECRETUL unui SUCCES

Vă voi împărtăși gânduri "frumoase" care s-au născut pe parcursul *primei conferințe naționale I*AERN* desfășurată la Cluj în zilele de 13-14 noiembrie. Am văzut lucruri deosebite, *realizate de copii*, îndrumați de profesori care au înțeles *importanța comunicației*... și nu doar atât.

"Am înțeles posibilitatea extraordinară pe care o oferă Internet-ul în a trece granițe fără să străbați distanțele efectiv, în a aduna copii din țări îndepărtate geografic, de a schimba idei și informații, de a realiza un pas înainte spre o ***lume mai bună****. Am observat că elevii devin profund motivați în a învăța, devenind mai puternici și mai încrezători în propriile forțe." (Cornelia Platon)*

Acum câțiva ani a sosit o invitație și la Liceul de Informatică din Cluj, de a participa la acțiunile I*AERN, dar după câteva încercări modeste, interesul a încetat. Dar în România au fost găsite resurse materiale și umane în localități și în școli despre care până acum am auzit mai puțin: ***Școala Generală nr.9 din Focșani, Școala Generală nr.10 din aceeași localitate, Grupul Școlar Eforie Sud, Liceul Știrbei Vodă din Călărași și nu în ultimul rând Liceul "Papiu Ilarian" din Dej, sunt în fruntea acțiunilor I*AERN.***

Poate ați vrea să mă întrebați: care este totuși legătura cu titlul acestui editorial? Nu putem sta inerți în așteptarea schimbărilor miraculoase pe care le-ar aduce o reformă (a educației și a învățământului). Reforma trebuie s-o facem noi. Să căutăm, să inventăm și să răspundem la mâna întinsă de instrumentarul bogat, oferit de *tehnologia informației*. Veți spune că nu este ușor, și că această tehnologie poate pătrunde rapid doar acolo unde sunt bani. Sper să vă conving că nu este așa.

Dar ce este I*AERN?

Acum 10 ani, în 1988 "perestroika" din fosta URSS a permis ca, sub forma unei experiențe, copii din câte o școală din Statele Unite și Moscova să comunice între ei prin intermediul Internet-ului. Și nu doar să comunice, ci să conlucreze la dezvoltarea unor pagini Web pe diverse teme care îi interesau. Experimentul a atras 12 școli, apoi s-a extins în 9 țări. Azi sunt înființate centre I*AERN în 50 de țări.

I*AERN este un program internațional, non-profit. Participanții colaborează la proiecte educaționale, formând o comunitate globală a elevilor, profesorilor și organizațiilor de tineret, angajați în folosirea telecomunicației, ca parte integrantă a procesului instructiv-educativ. Este o oportunitate excepțională pentru participanți de a concepe propriul proiect și de a găsi parteneri în întreaga lume. Singura "obligație" pe care trebuie s-o respecte fiecare participant este de a furniza informații *corecte*.

România în I*AERN

Cu toate că România participă activ la proiectele I*AERN-ului și obține rezultate deosebite cu ocazia diferitelor conferințe (clasice și electronice), respectiv concursuri, deocamdată nu există nici o organizație implicată în susținerea financiară a programului. Sprijinul Ministerului Educației se reduce la unul moral; Romtelecom-ul nu facilitează conectarea la Internet, iar Fundația Soros pentru o Societate Deschisă nu va putea finanța până la nesfârșit un asemenea program care sperăm să cuprindă foarte multe școli.

Este regretabil că Ministerul Educației nu găsește fonduri pentru sprijinirea unui asemenea program care învață copiii, de la cele mai fragede vârste, să *caute* informații *corecte*, să cerceteze și să creeze jucându-se, într-o atmosferă entuziasmantă, fără efortul neagreat, tipic sistemului clasic de învățare.

Știm. Nu sunt bani. Dar dacă mă gândesc bine, și cei puțini, câți ar fi, se împart fără să se țină cont de prioritățile naționale de importanță vitală.

Poți să participi și Tu la I*AERN!

D-na profesoara Cornelia Platon, de la Liceul "Papiu Ilarian" din Dej, cea care, împreună cu elevii săi, a însuflețit acest program printr-o lăudabilă contribuție, vă va răspunde la orice întrebare. Poate fi contactată pe e-mail la adresa: nelly@lcjdap.soroscj.ro.

Clara Ionescu

SOLUȚIILE problemelor propuse

Între 26 iulie - 9 august la Constanța s-a desfășurat a doua tabără de antrenament a lotului olimpic național de informatică. La sfârșitul celei de a doua săptămâni membrii lotului lărgit au primit șapte probleme. Vă prezentăm în continuare soluțiile problemelor.

P079801: Coloane

Problemă propusă de conf. dr. Henri Luchian, Universitatea "Al. I. Cuza", Iași
Soluția îi aparține lui Mihai Scorțaru, student, Universitatea Tehnică, Cluj

Deoarece pentru această problemă nu se cunoaște o soluție polinomială, algoritmul propus pentru rezolvare este unul genetic. Rezultă că metoda de rezolvare va fi una nedeterministă, negarantându-se obținerea unui rezultat corect la fiecare rulare.

Pentru început vom observa că nu există soluție în cazul în care o anumită linie conține numai zerouri. În cazul în care nu există o asemenea linie, prin selectarea tuturor coloanelor sigur se obține o soluție.

Pentru implementarea algoritmului genetic vom folosi o populație de zece cromozomi, probabilitatea de mutație va fi 0.2 și va crește cu 10% la fiecare 100 de generații, iar probabilitatea de recombinare va fi 0.6.

Un cromozom va conține `n` gene (`n` = numărul de coloane), fiecare genă putând avea valoarea `0` (caz în care coloana corespunzătoare nu trebuie aleasă) sau `1` (caz în care coloana corespunzătoare trebuie aleasă).

Pentru fiecare cromozom trebuie calculat un punctaj care să-i caracterizeze adaptarea. Acest punctaj va fi dat de formula: `P=4*n+1-c-2*l-x*n`, unde:

`N` reprezintă numărul de coloane ale matricei
`C` reprezintă numărul de coloane selectate
`L` reprezintă numărul de linii pentru care suma elementelor de pe linie este 0
`X=0` dacă `L=0` și `X=1` dacă `L≠0`;

Pentru fiecare generație trebuie să realizăm o selecție, să efectuăm câteva mutații și să recombinăm cromozomii.

Selecția se realizează astfel:

- se determină punctajul fiecărui cromozom;
- se calculează suma `S` a punctajelor cromozomilor;
- se calculează probabilitatea de selecție a fiecărui cromozom; aceasta va fi egală cu raportul dintre punctajul său și valoarea `S`;
- se împarte intervalul `[0,1)` în zece subintervale astfel încât lungimea intervalului corespunzător unui cromozom să fie egală cu probabilitatea de selecție a acelui cromozom; de exemplu, pentru o populație care are cromozomii 1, 3, 5, 7, 9 cu probabilitatea de selecție 0.15 și cromozomii 2, 4, 6, 8, 10 cu probabilitatea de selecție 0.05, intervalele corespunzătoare vor fi:
`[0,0.15)`, `[0.15,0.2)`, `[0.2,0.35)`, `[0.35,0.4)`, `[0.4,0.55)`, `[0.55,0.6)`, `[0.6,0.75)`, `[0.75,0.8)`, `[0.8,0.95)`, `[0.95,1)`;
- se generează zece numere aleatoare subunitare și pentru fiecare din aceste numere se determină intervalul din care face parte iar cromozomul corespunzător este selectat (se observă că un același cromozom poate fi ales de mai multe ori).

Pentru mutații vom proceda în felul următor:

- se generează câte un număr aleator subunitar pentru fiecare cromozom; dacă acest număr este mai mic decât probabilitatea de mutație atunci acest cromozom va deveni mutant;
- pentru a realiza mutația propiu-zisă se selectează o genă și i se schimbă valoarea.

Recombinarea va necesita următoarele operații:

- se generează câte un număr aleator subunitar pentru fiecare cromozom; dacă acest număr este mai mic decât probabilitatea de recombinare atunci acest cromozom va fi selectat pentru a fi recombinat;
- se aleg perechi de câte doi cromozomi din lista celor selectați pentru recombinare;
- pentru fiecare pereche de cromozomi se aleg două numere aleatoare întregi `P1` și `P2` (`P1 < P2`) din intervalul `[1,n]` care vor indica faptul că fiecare cromozom își va păstra primele `P1` și ultimele `n-P2` gene;
- se copiază din cromozomul pereche informațiile cores-

punzătoare celorlalte gene;
- noii cromozomi își vor înlocui "părinții" în populație.

La fiecare generație cel mai bine adaptat cromozom va fi păstrat chiar dacă, în urma selecției, mutației sau recombinării, el nu ar mai trebui să facă parte din populație.

Listing COLOANE.PAS

```
Program Coloane_Algoritm_Genetic;
uses Crt;

const popsize=10;    { numarul de cromozomi }
      ppm=0.2;
      { probabilitatea initiala de mutatie }
      pc=0.6;{probabilitatea de recombinare}
type cromozom=array[1..200] of Byte;

var a:array[1..200,1..200] of Byte;
    f:Text;
    m,n:Byte; gen:Longint;
    pp1,pp,pop:array[1..popsize] of cromozom;
    p1,p2,sol,testat:cromozom;
    c:array[1..200]of Boolean;
    c1:array[1..200]of Integer;
    c2:array[0..200]of Real;
    nri1,nri,nrn,maxim:Integer;
    pm:Real;

procedure citire;
var i,j:Byte;
begin
  Assign(f,'COLOANE.IN'); Reset(f);
  Readln(f,m,n);
  for i:=1 to m do
    for j:=1 to n do Read(f,a[i][j]);
  Close(f)
end;

procedure farasolutie;
begin
  Assign(f,'COLOANE.OUT'); Rewrite(f);
  Writeln(f,0);
  Close(f);
  Halt
end;

procedure verificare;
var i,j,nr:Byte;
begin { se verifica existenta unei solutii }
  for i:=1 to m do
  begin
    nr:=0;
    for j:=1 to n do nr:=nr+a[i][j];
    if nr=0 then farasolutie
  end
end;
```

```
function punctaj: integer;
var i,j:Byte;
    pt:Integer;
    da:Boolean;
begin
  Fillchar(c,Sizeof(c),false);
  pt:=4*n+1;
   { se scade numarul de coloane selectate }
  for i:=1 to n do
    if testat[i]=1
    then
    begin
      Dec(pt);
      for j:=1 to m do
        if a[j][i]=1 then c[j]:=true
    end;
  da:= false;
   { se scade de doua ori numarul de linii }
   { pentru care suma elementelor de pe    }
   { linie este 0                          }
  for i:=1 to m do
    if not c[i]
    then
    begin
      Dec(pt,2); da:=true
    end;
     { in cazul in care a existat cel      }
     { putin o linie pentru care suma      }
     { elementelor a fost 0 se mai scade n }
  if da then Dec(pt, n);
  punctaj:=pt
end;

procedure selectie;
var i,j:Byte; x:Real;
begin
  for i:=1 to popsize do
  begin          { se calculeaza punctajele }
    testat:=pop[i]; c1[i]:=punctaj
  end;
  c2[0]:=0;
 { se determina probabilitatea de selectie }
  for i:=1 to popsize do
    c2[i]:=c2[i-1]+c1[i];
  for i:=1 to popsize do
    c2[i]:=c2[i]/c2[popsize];
                { se selecteaza cromozomii }
  for i:=1 to popsize do
  begin
    x:=Random;
    j:=1;
    while c2[j]<x do Inc(j);
    pp[i]:=pop[j]
  end;
  pop:=pp
end;
```

```
procedure mutant(x:Byte);
var poz:Byte;
begin
    { se alege gena care va fi, modificata }
  poz:=Random(n)+1;
                          { se modifica gena }
  pop[x][poz]:=1-pop[x][poz];
    { se tasteaza adaptarea noului cromozom}
  testat:=pop[x];
  if punctaj>maxim
  then
  begin maxim:=punctaj; sol:=testat end
end;

procedure mutatie;
var i:Byte;
begin
  for i:=1 to popsize do
    if pm>Random then mutant(i)
end;

procedure imperecheaza;
var i,poz1,poz2:Byte;
begin       { se determina pozitiile intre }
            { care vor fi schimbate genele }
  poz1:=Random(n-2)+1;
  poz2:=Random(n-2)+1;
  if poz1>poz2
  then
  begin
  { poz1 trebuie sa fie mai mic decat poz2 }
    poz1:=poz1 xor poz2;
    poz2:=poz1 xor poz2;
    poz1:=poz1 xor poz2
  end;
                        { se modifica genele }
  for i:=poz1+1 to poz2 do
  begin
    p1[i]:=p1[i] xor p2[i];
    p2[i]:=p1[i] xor p2[i];
    p1[i]:=p1[i] xor p2[i]
  end
end;

procedure recombina;
var a,i,x:Byte;
begin
  nri1:=nri;
  x:=nri div 2;
  i:=1;
  while i<=x do
  begin
    { se selecteaza o pereche de cromozomi }
    a:=Random(nri)+1;
    Dec(nri);
    p1:=pp[a];
    pp[a]:=pp[nri+1];
    a:=Random(nri)+1;
    Dec(nri);
    p2:=pp[a];
    pp[a]:=pp[nri+1];
    Inc(i);
    imperecheaza;
  { cromozomii nou obtinuti isi inlocuiesc }
                  { parintii in populatie }
    pp[nri+1]:=p1;
    pp[nri+2]:=p2;
            { se testeaza adaptabilitatea }
                    { noilor cromozomi }
    testat:=p1;
    if punctaj>maxim
    then
    begin maxim:=punctaj; sol:=testat end;
    testat:=p2;
    if punctaj>maxim
    then
    begin maxim:=punctaj; sol:=testat end
  end
end;

procedure incrucisare;
var i:Byte;
begin
  nri:=0;
  nrn:=0;
              { se selecteaza cromozomii }
              { care vor fi recombinati  }
  for i:=1 to popsize do
    if Random<pc
    then
    begin
      Inc(nri); pp[nri]:=pop[i]
    end
    else
    begin
      Inc(nrn); pp1[nrn]:=pop[i]
    end;
  recombina
end;

procedure generatie;
var i:Byte;
begin
  selectie;
  mutatie;
  incrucisare;
              { se creeaza noua populatie }
  pop:=pp;
  for i:=nri1+1 to popsize do
    pop[i]:=pp1[i-nri1+1];
  pop[Random(popsize)+1]:=sol
end;
```

```
procedure initializare;
var i,j:Byte;
begin      { se genereaza aleator structura }
                  { primilor noua cromozomi }
  for i:=1 to popsize - 1 do
    for j:=1 to n do
      pop[i][j]:=Random(2);
          { al zecelea cromozom va fi cel   }
          { corespunzator cazului in care   }
          { toate coloanele ar fi selectate }
  for j:=1 to n do pop[popsize][j]:=1;
{ se testeaza adaptabilitatea cromozomilor }
  for i:=1 to popsize do
  begin
    testat:=pop[i];
    if punctaj>maxim
    then
    begin
      maxim:=punctaj;
      sol:=testat
    end
  end
end;

procedure afisare;
var i,nr:Byte;
begin
  Assign(f,'coloane.out'); Rewrite(f);
  nr:=0;
          { se numara coloanele selectate }
  for i:=1 to n do Inc(nr,sol[i]);
  Writeln(f,nr);
  for i:=1 to n do
    if sol[i]=1 then Write(f,i,' ');
  Writeln(f);
  Close(f)
end;

Begin
  Clrscr;
  Randomize;
  citire;
  verificare;
  initializare;
  pm:=ppm;
  gen:=1;
  repeat
    if gen mod 100=0
    then pm:=pm*1.1;
        { creste probabilitatea de mutatie  }
    generatie;
    Inc(gen)
  until Keypressed; { executia se termina   }
  afisare          { la apasarea unei taste }
End.
```

P079802: Cuburi

Soluție propusă de autorii problemei,
prof. Maria și Adrian Niță,
Liceul "Emanoil Gojdu", Oradea

Metoda de rezolvare "reface" cuburile și verifică dacă cele două cuburi date sunt identice. În acest scop unul din cuburi este "rotit" astfel încât fețele superioare ale celor două cuburi să coincidă, apoi se verifică dacă și restul fețelor coincid.

Listing: CUBURI.CPP

```
#include <stdio.h>
int adj[6][4]={4,3,2,1,0,2,5,4,0,3,5,1,0,4,
               5,2,0,1,5,3,2,3,4,1};
char cub1[10], cub2[10];
FILE *in,*out;
int cubcmp(void);
main(){
  in=fopen("cub.in","rt");
  out=fopen("cub.out","wt");
  while (fscanf(in,"%s %s",cub1,cub2)!=EOF)
  {
    if (cubcmp()) fprintf(out,"DA\n");
    else fprintf(out,"NU\n");
  }
  fclose(in);
  fclose(out);
  return(0);
}

int cubcmp(void)
{
  int ind,i,j,len;
  for(ind=0; cub1[0]!=cub2[ind]; ind++);
  for(i=j=0; j<4 && cub1[adj[0][0]]!=
                   cub2[adj[ind][j]]; j++);
    if (j==4) return(0);
  for(len=0; len<4; len++, i++, j++)
    if (cub1[adj[0][i % 4]]!=
          cub2[adj[ind][j % 4]]) return (0);
  return (1);
}
```

P079803: Eschimoși

Problemă propusă de lect. Clara Ionescu,
și rezolvată de conf. dr. Teodor Toadere
Universitatea "Babeș Bolyai", Cluj

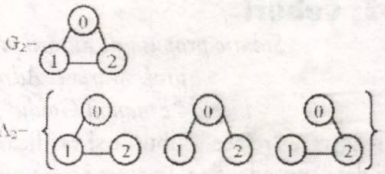
Fie $G_n = (X, U)$ cu $X = \{0, 1, \ldots, n\}$ și
$U = \{(i, j) \mid i, j \in X, i = 0 \text{ sau } i - j = 1\}$

Notăm cu $A_n = \{ P \mid P = (X, V), V \subset U \text{ și } P \text{ arbore}\}$ și cu $|A_n| = a_n$. Dorim să determinăm o relație de recurență pentru șirul $(a_n)_{n \in N}$.

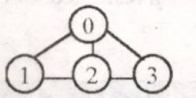
Mai întâi să constatăm că:

a) $a_1 = 1$ pentru că $A_1 = \{G_1\}$

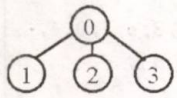
b) $a_2 = 3$ pentru că:

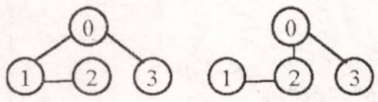


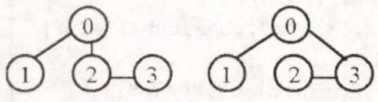
c) $|a_3| = 8$ pentru că arborii de acoperire a lui G_3, adică

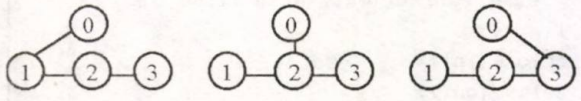


sunt









Tot prin enumerarea arborilor lui G_4 se obține $a_4 = 21$

Pentru a exprima pe a_n facem următoarea descompunere (partiționare) de mulțimi

$A_n = E \cup F \cup G$, unde

E = mulțimea arborilor din A_n care conțin muchia $(0, n)$ și nu conțin muchia $(n, n-1)$;

F = mulțimea arborilor din A_n care conțin muchia $(n, n-1)$ și nu conțin muchia $(0, n)$;

G = mulțimea arborilor din A_n care conțin atât muchia $(0, n)$ cât și muchia $(n, n-1)$.

Se deduce că $E \cap F = E \cap G = F \cap G = \varnothing$ și deci

$a_n = |A_n| = |E| + |F| + |G|$

Vom demonstra că

a) $|E| = |A_{n-1}| = a_{n-1}$

b) $|F| = |A_{n-1}| = a_1$

c) $|G| = \left|\bigcup_{k=1}^{n-2} A_k\right| = \sum_{k=1}^{n-2} a_k$

prin a preciza niște bijecții între mulțimile respective.

Astfel:

a) $f: E \to A_{n-1}$, unde pentru fiecare $P \in E$, $f(P)$ este graful obținut din P prin eliminarea vârfului n și a muchiei $(0, n)$. Din definiție deducem că $f(P)$ are $n-1$ vârfuri este conex și are $n-2$ muchii deci $f(P) \in A_{n-1}$. Se verifică ușor că f este injectivă ($\forall P_1 \neq P_2$ din $E \Rightarrow f(P_1) \neq f(P_2)$) și că f este surjectivă ($\forall P \in A_{n-1}$ prin adăugarea vârfului n și a muchiei $(0, n)$ se obține arborele P' cu $P' \in E$ și $f(P') = P$).

b) analog, $h: F \to A_{n-1}$, definită prin $h(P)$ este graful obținut din arborele $P \in F$ prin eliminarea vârfului n și a muchiei $(n, n-1)$, este o bijecție

c) pentru demonstrarea egalității $|G| = \left|\bigcup_{k=0}^{n-2} A_k\right|$ considerăm funcția $g: \bigcup_{k=0}^{n-2} A_k \to G$,

prin $\forall k \in \{0,\ldots,n-2\}$ și $\forall P \in A_k$ definim pe $g(P)$ ca fiind graful obținut din P prin adăugarea lanțului $\{k+1,\ldots,n-1, n, 0\}$ și a vârfurilor corespunzătoare (deci între vârfurile k și $k+1$ în $g(P)$ nu există muchie. Se demonstrează ușor că $g(P)$ (este arbore din G_n care conține muchiile $(n-1, n)$ și $(n, 0)$) și că g este o bijecție (din $P_1 \neq P_2 \Rightarrow g(P_1) \neq g(P_2)$ și că $\forall P \in G\ \exists k$ astfel încât $\forall i \in \{k+1, \ldots, n\}$, $(0, i) \notin P$ și deci P este format din lanțul $\{k+1, \ldots, n, 0\}$ și un arbore al lui G_k.

Prin urmare

$$a_n = |E| + |F| + |G| = 2a_{n-1} +$$

$$+ \sum_{k=1}^{n-2} a_k = 2a_{n-1} - a_{n-2} + (2a_{n-2} + \sum_{k=1}^{n-1} a_k) = 2a_{n-1} - a_{n-2} + a_{n-1}$$

Deci, relația de recurență este: $a_n = 3a_{n-1} - a_{n-2}$

Cum rădăcinile ecuației caracteristice $x^2 - 3x + 1 = 0$ sunt

$x_{1,2} = \frac{3 \pm \sqrt{5}}{2}$, expresia termenului general este:

$$a_n = c_1 \left(\frac{3+\sqrt{5}}{2}\right)^n + c_2 \left(\frac{3-\sqrt{5}}{2}\right)^n .$$

Constantele c_1 și c_2 se obțin din sistemul $a_1 = 1$ și $a_2 = 8$ și sunt

$$c_1 = -c_2 = \frac{1}{\sqrt{5}}$$

Deci

$$a_n = \frac{1}{\sqrt{5}} \left[\left(\frac{3+\sqrt{5}}{2}\right)^n - \left(\frac{3-\sqrt{5}}{2}\right)^n \right] =$$

$$= \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2}\right)^{(2n-1)+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{(2n-1)+1} \right].$$

Cum se știe că termenii șirului lui Fibonacci sunt

$$f_n = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1} \right]$$

constatăm că $a_n = f_{2n-1}\ \forall n \in \mathbb{N}^*$.

Implementarea, deosebit de simplă, conține totuși un "apel" la cunoștințe de programare: pentru a calcula și afișa numărul cerut (poate avea 15 cifre!), acesta trebuie să fie declarat de tipul `Comp`. Pentru a lucra cu acest tip, este nevoie ca opțiunea de compilare *Numeric Processing 80x87* să fie activată. Activarea se realizează folosind directiva de compilare `{$N+}`.

Listing ESCHIMOSI.PAS

```
{$N+}
Program Eschimos;
var n,i:Byte;
    a,b,c:Comp;
    fi,fo:Text;
Begin
  Assign(fi,'ESC.IN'); Reset(fi);
  Readln(fi,n); Close(fi);
  Assign(fo,'ESC.OUT'); Rewrite(fo);
  a:=0; b:=1; c:=a+b;
  for i:=2 to n do
  begin
    a:=b; b:=c; c:=a+b;
    a:=b; b:=c; c:=a+b
  end;
  Writeln(fo,c:15:0); Close(fo)
End.
```

P079804: Rezistențe

Problemă propusă și rezolvată de Valentin Gheorghiță, student, Universitatea Politehnică, București

Primul lucru delicat în rezolvarea acestei probleme este structurarea datelor. Pentru a putea păstra informațiile despre fiecare rezistență vom crea o listă cu N (numărul de rezistențe la un moment dat) înregistrări, pentru fiecare rezistență păstrând nodurile i și j (i<j) între care se află rezistența și valoarea rezistenței.

În continuare vom crea cele trei proceduri ce tratează transformările prezentate (serie, paralel, triunghi-stea) în problemă. Totdeauna, în momentul apelării uneia dintre aceste proceduri lista cu rezistențe va fi sortată crescător în funcție de i (nodul cu valoarea cea mai mică de care este legată rezistența) iar în caz de egalitate în funcție de j (nodul cu valoarea cea mai mare de care este legată rezistența). Sortarea se va face utilizând algoritmul qsort, complexitatea lui medie fiind $n*\log_2 n$.

Procedura de legare în paralel are complexitatea liniară, egală cu numărul de rezistențe aflate în circuit. În procedură se va parcurge lista cu rezistențe, iar dacă două sau mai multe rezistențe se află între aceleași două noduri, atunci acele rezistențe vor fi eliminate din listă și în locul lor se va adăuga o singură rezistență aflată între aceleași noduri, dar cu valoare calculată după formula dată în problemă.

Pentru transformarea triunghi-stea se va defini gradul unui nod ca fiind numărul de rezistențe care sunt legate de acel nod. În continuare, în procedura care rezolvă această transformare, pentru fiecare două rezistențe aflate între nodurile i și j, respectiv i și k (adică au un nod comun), dacă nodurile i, j, k au gradul mai mare decât 2 se va verifica dacă există o rezistență între nodurile j și k. Dacă există, atunci se vor elimina cele trei rezistențe, introducându-se în locul lor alte trei cu valori calculate după formulele date, legate la un capăt de un nod nou introdus, iar la celălalt de nodurile i, j, respectiv k.

Pentru legarea în serie, se observă că, dacă un nod are gradul doi și nu este unul dintre nodurile între care trebuie calculată rezistența echivalentă, atunci el poate fi eliminat împreună cu rezistențele care se leagă de el. În cazul găsirii unui astfel de nod se parcurge recursiv circuitul într-o parte și în cealaltă, până se va ajunge la noduri care au gradul diferit de 2. Rezistențele și nodurile (cu gradul egal cu 2) prin care s-a trecut în această parcurgere, sunt eliminate, introducându-se o nouă rezistență care are ca valoare suma rezistențelor eliminate și este legată de cele două noduri în care ne-am oprit cu parcurgerea.

În final se vor utiliza procedurile definite pentru a calcula rezistența echivalentă. Pentru aceasta, vom repeta următoarea secvență până când lista rezistențelor va conține o singură înregistrare care este chiar rezistența echivalentă ce trebuia calculată:

- sortare
- procedura legare în paralel
- sortare
- procedura transformare triunghi-stea
- sortare
- procedura legare în serie

Observație: rezultatul se cere cu trei zecimale exacte, deci trebuie puțină atenție la afișarea cu autoformatare, pentru că de exemplu, în Pascal, aceasta produce rotunjirea și nu trunchierea rezultatului.

Listing: REZISTENTE.PAS

```
{$M 65000,0,655360}
program rezistenta_echivalenta;
const ni='REZ.IN';
      no='REZ.OUT';
type sir=array[1..10000] of Integer;
     psir=^sir;
     vect=array[1..10000] of Real;
     pvect=^vect;
var grad,start,fin,a,b,aa,bb,s,v1,v2:psir;
    v, vv:pvect;
    f:Text;
    n,m,nm,i,j,x,y,pz,med,li,ls:Integer;
    aux,nod,nnod,ad,capat,inceput:Integer;
    sfarsit,p,sursa,dest:Integer;
    r,suma,k,auxr:Real;

procedure load;
begin
  New(grad); New(start);
  New(fin); New(s);
  New(a); New(b);
  New(v); New(aa);
  New(bb); New(vv);
  New(v1); New(v2);
  Assign(f,ni); Reset(f);
  Readln(f,n);
  Readln(f,sursa,dest);
  m:=0;
```

```
  while not Seekeof(f) do
  begin
    Readln(f,i,j,k);
    if i>j
    then
    begin
      aux:=i; i:=j; j:=aux
    end;
    m:=m+1;
    a^[m]:=i;
    b^[m]:=j;
    v^[m]:=k
  end;
  Close(f);
  m:=m+1;
  a^[m]:=sursa;
  b^[m]:=n+1;
  v^[m]:=0;
  m:=m+1;
  a^[m]:=dest;
  b^[m]:=n+2;
  v^[m]:=0;
  n:=n+2
end;

procedure qsort(li,ls:Integer);
var i,med:Integer;
begin
  i:=li;
  j:=ls;
  med:=a^[(li+ls) shr 1];
  repeat
    while a^[i]<med do i:=i+1;
    while a^[j]>med do j:=j-1;
    if i<=j
    then
    begin
      aux:=a^[i]; a^[i]:=a^[j]; a^[j]:=aux;
      aux:=b^[i]; b^[i]:=b^[j]; b^[j]:=aux;
      auxr:=v^[i]; v^[i]:=v^[j]; v^[j]:=auxr;
      i:=i+1;
      j:=j-1
    end
  until i>j;
  if li<j
  then qsort(li,j);
  if i<ls
  then qsort(i,ls)
end;

procedure qqsort(li,ls:Integer);
var i,j,med:Integer;
begin
  i:=li;
  j:=ls;
  med:=b^[(li+ls) shr 1];
  repeat
    while b^[i]<med do i:=i+1;
    while b^[j]>med do j:=j-1;
    if i<=j
    then
    begin
      aux:=a^[i]; a^[i]:=a^[j]; a^[j]:=aux;
      aux:=b^[i]; b^[i]:=b^[j]; b^[j]:=aux;
      auxr:=v^[i]; v^[i]:=v^[j]; v^[j]:=auxr;
      i:=i+1;
      j:=j-1
    end
  until i>j;
  if li<j
  then qqsort(li,j);
  if i<ls
  then qqsort(i,ls)
end;

procedure sort;
begin
  qsort(1,m);
  x:=1;
  while x<=m do
  begin
    y:=x;
    while (a^[y]=a^[x]) and (y<=m) do y:=y+1;
    qqsort(x,y-1);
    x:=y
  end
end;

procedure paralel;
begin
  nm:=0;
  x:=1;
  while x<=m do
  begin
    y:=x;
    while (a^[y]=a^[x]) and (b^[y]=b^[x])
                        and (y<=m) do
      y:=y+1;
    if y=x+1
    then r:=v^[x]
    else
      begin
        r:=0;
        for i:=x to y-1 do r:=r+1/v^[i];
        r:=1/r
      end;
    nm:=nm+1;
    aa^[nm]:=a^[x];
    bb^[nm]:=b^[x];
    vv^[nm]:=r;
    x:=y
  end;
```

```
  m:=nm;
  for i:=1 to m do
  begin
    a^[i]:=aa^[i];
    b^[i]:=bb^[i];
    v^[i]:=vv^[i]
  end
end;

procedure det;
begin
  for i:=1 to n do
  begin
    start^[i]:=0;
    fin^[i]:=0
  end;
  x:=1;
  while x<=m do
  begin
    y:=x;
    while (a^[y]=a^[x]) and (y<=m) do
      y:=y+1;
    nod:=a^[x];
    start^[nod]:=x;
    fin^[nod]:=y-1;
    x:=y
  end;
  for i:=1 to n do
    grad^[i]:=0;
  for i:=1 to m do
  begin
    nod:=a^[i];
    nnod:=b^[i];
    grad^[nod]:=grad^[nod]+1;
    grad^[nnod]:=grad^[nnod]+1
  end
end;

procedure cauta(li,ls:Integer);
begin
  if li<=ls
  then
  begin
    med:=(li+ls) shr 1;
    if nnod=b^[med]
    then pz:=med
    else
      if nnod<b^[med]
      then cauta(li,med-1)
      else cauta(med+1,ls)
  end
end;

procedure tr_stea;
begin
 for i:=1 to m do s^[i]:=0;
ad:=0;
nm:=0;
for i:=1 to n do
 if grad^[i]>2
 then
   if not((start^[i]=0) and (fin^[i]=0))
   then
     for x:=start^[i] to fin^[i]-1 do
       if s^[x]=0
       then
         for y:=x+1 to fin^[i] do
           if s^[y]=0
           then
           begin
             nod:=b^[x];
             nnod:=b^[y];
             if (grad^[nod]>2) and
                              (grad^[nnod]>2)
             then
             if not((start^[nod]=0) and
                           (fin^[nnod]=0))
             then
             begin
               li:=start^[nod];
               ls:=fin^[nod];
               pz:=0;
               cauta(li,ls);
               if pz<>0
               then
                if s^[pz]=0
                then
                begin
                  suma:=v^[x]+v^[y]+v^[pz];
                  ad:=ad+1;
                  nm:=nm+1;
                  aa^[nm]:=i;
                  bb^[nm]:=n+ad;
                vv^[nm]:=(v^[x]*v^[y])/suma;
                  nm:=nm+1;
                  aa^[nm]:=nod;
                  bb^[nm]:=n+ad;
                vv^[nm]:=(v^[x]*v^[pz])/suma;
                  nm:=nm+1;
                  aa^[nm]:=nnod;
                  bb^[nm]:=n+ad;
                vv^[nm]:=(v^[y]*v^[pz])/suma;
                  s^[x]:=1;
                  s^[y]:=1;
                  s^[pz]:=1;
                  y:=fin^[i]
                end
             end
           end;
n:=n+ad;
for i:=1 to m do
 if s^[i]=0
```

```
  then
  begin
   nm:=nm+1;
   aa^[nm]:=a^[i];
   bb^[nm]:=b^[i];
   vv^[nm]:=v^[i]
  end;
 for i:=1 to m do
 begin
  a^[i]:=aa^[i];
  b^[i]:=bb^[i];
  v^[i]:=vv^[i]
 end
end;

procedure rec(nod,tata:Integer);
var ch:Boolean;
begin
  nnod:=tata;
  ch:=false;
  if nod>nnod
  then
  begin
    aux:=nod; nod:=nnod; nnod:=aux;
    ch:=true
  end;
  li:=start^[nod];
  ls:=fin^[nod];
  cauta(li,ls);
  s^[pz]:=1;
  suma:=suma+v^[pz];
  if ch
  then
  begin
    aux:=nod; nod:=nnod; nnod:=aux
  end;
  if grad^[nod]=2
  then
    if v1^[nod]=tata
    then rec(v2^[nod],nod)
    else rec(v1^[nod],nod)
  else capat:=nod
end;

procedure serie;
begin
  for i:=1 to n do
  begin v1^[i]:=0; v2^[i]:=0 end;
  for i:=1 to m do
  begin
    nod:=a^[i]; nnod:=b^[i];
    if grad^[nod]=2
    then
      if v1^[nod]=0
      then v1^[nod]:=nnod
      else v2^[nod]:=nnod;
```

```
    if grad^[nnod]=2
    then
      if v1^[nnod]=0 then v1^[nnod]:=nod
                     else v2^[nnod]:=nod
  end;
  for i:=1 to m do
    s^[i]:=0;
  nm:=0;
  for i:=1 to m do
    if s^[i]=0
    then
    begin
      nod:=a^[i]; nnod:=b^[i]; p:=0;
      if grad^[nod]=2 then p:=nod;
      if grad^[nnod]=2 then p:=nnod;
      if p<>0
      then
      begin
        suma:=0;
        rec(v1^[p],p);
        inceput:=capat;
        rec(v2^[p],p);
        sfarsit:=capat;
        nm:=nm+1;
        if inceput>sfarsit
        then
        begin
          aux:=inceput;
          inceput:=sfarsit;
          sfarsit:=aux
        end;
        aa^[nm]:=inceput; bb^[nm]:=sfarsit;
        vv^[nm]:=suma
      end
    end;
    for i:=1 to m do
      if s^[i]=0
      then
      begin
        nm:=nm+1;
        aa^[nm]:=a^[i]; bb^[nm]:=b^[i];
        vv^[nm]:=v^[i]
      end;
    m:=nm;
    for i:=1 to m do
    begin
      a^[i]:=aa^[i]; b^[i]:=bb^[i];
      v^[i]:=vv^[i]
    end
end;
procedure scrie;
begin
  Assign(f,no); Rewrite(f);
  Writeln(f,Trunc(v^[1]*1000)/1000:0:3);
  Close(f)
end;
```

```
Begin
  load;
  while m>1 do
  begin
    sort;
    paralel;
    det;
    tr_stea;
    sort;
    det;
    serie
  end;
  scrie
End.
```

P079805: Seiful

Problemă propusă și rezolvată de prof. Dana Lica, Liceul "I. L. Caragiale", Ploiești

Pentru orice grup care trebuie format din n-2 persone (conform cerințelor problemei) trebuie să existe cel puțin un lacăt pe care nu-l va deschide nimeni din grup. Rezultă că celelalte două persoane rămase în afara grupului dețin fiecare cheia lacătului respectiv. Astfel, atunci când una din cele două persoane intră în grupul de n-2 persoane, lacătul va putea fi deschis.

Deducem că la fiecare lacăt trebuie să existe exact două chei. Tot de aici se deduce că numărul de lacăte este egal cu numărul grupurilor distincte formate din n-2 persoane, adică n(n-1)/2.

Atunci numărul total de chei va fi n(n-1).

Listing: LACATE.PAS

```
program Chei_si_Lacate;
var n:Longint;

procedure citeste;
var f:Text;
begin
  Assign(f,'SEIF.IN'); Reset(f);
  Readln(f,n); Close(f)
end;

procedure scrie;
var f:Text; i,j:Longint;
begin
  Assign(f,'SEIF.OUT'); Rewrite(f);
  Writeln(f,n*(n-1) shr 1,' ',n*(n-1));
  for i:=1 to n-1 do
    for j:=i+1 to n do Writeln(f,i,' ',j);
  Close(f)
end;

Begin
  citeste;
  scrie
End.
```

P079806: Piramida lui Keops

Problemă propusă și rezolvată de Marius Vlad, student, Universitatea Politehnică, București

Deoarece problema cere determinarea unui *minim* care ar trebui căutat în mulțimea unor soluții posibile care se pot genera arborescent, s-a ales ca metodă de rezolvare metoda programării dinamice.

Vom calcula cu ajutorul unei proceduri recursive (AjungeIn) drumurile minime de la fiecare bloc de pe o latură exterioară a bazei piramidei la celelalte blocuri ale bazei.

Pentru fiecare nivel (diferit de bază) și pentru fiecare bloc în parte de pe acest nivel, presupunem că ajungem în acel bloc venind de pe nivelul inferior și calculăm cu ajutorul aceleiași proceduri recursive lungimea drumului plecând de la blocul respectiv la toate blocurile existente pe acel nivel.

Procedura recursivă AjungeIn este asemănătoare cu o procedură *fill* obișnuită și poate fi rapid implementată în timp de concurs. Se oprește în momentul în care lungimea drumului pe calea urmată este mai mare decât lungimea drumului minim găsit.

De fiecare dată când se găsește un drum de lungime mai mică decât cel existent se salvează adresa blocului precedent pentru a reconstrui drumul de întoarcere plecând de la blocul din vârf.

Listing PIRAMIDA.PAS

```
{$M 65500,0,655360}
program Piramida;
const DIMAX=4500; {Lungime maxima a
   vectorului in care se salveaza piramida }
      NMAX=30;
       { Lungimea maxima a bazei piramidei }
type piramida_byte=array[0..DIMAX] of Byte;
     piramida_long=array[0..DIMAX] of Longint;

var i,j,k,n,m:Integer;
    d,              { Duritatile blocurilor }
    hback,lback,cback:piramida_byte;
                { Adresa blocului precedent }
    min:piramida_long;
       { Lungimea minima pana la acel bloc }
    pos:array[0..NMAX] of Longint;
    w:array[0..NMAX] of Integer;
    fin,fout:Text;

procedure Initializare;
{ Initializeaza variabile, precalculeaza date }
var s,k:Longint;
begin
  m:=(n div 2)+1;   { Inaltimea piramidei    }
  w[0]:=n;          { Dimensiunea bazei      }
  pos[0]:=0;        { Pozitia de inceput in  }
                    { vector a bazei         }
  s:=0;
```

```
  for k:=1 to m-1 do
  begin
    w[k]:=w[k-1]-2;{Dimensiunea nivelului k}
    s:=s+Longint(w[k-1]*w[k-1]);
    pos[k]:=s      { Pozitia de inceput in }
                   { vector a nivelului k  }
  end;
  for k:=0 to DIMAX-1 do min[k]:=MaxLongInt
      { Initializare vector lungimi gasite }
end;

function Offset(h,l,c:Integer):Longint;
    { Afla pozitia in vector a blocului de }
    { piramida de la nivelul h, linia l si }
    { coloana c                            }
begin
  Offset:=Longint(pos[h])+Longint(l*w[h]+c)
end;

procedure AjungeIn(bh,bx,by,h,x,y:Integer;
                                  v:Longint);
var pos:Integer;
{ bh,bx,by - Adresa blocului vecin de unde }
                                { s-a plecat }
{ h,x,y - Adresa blocului din care se refac}
                                { drumurile }
{ v - Lungimea drumului pana la acel bloc
      inclusiv }
begin
  pos:=Offset(h,x,y);
  if v<min[pos]
  then
  begin   { Daca drumul gasit este mai mic }
    min[pos]:=v;{ Marchez lungimea drumului }
    hback[pos]:=bh;
    { Si adresa blocului din care am venit }
    lback[pos]:=bx;
{ pentru a gasi drumul minim plecand de la }
    cback[pos]:=by;          { blocul de sus }
        { Incerc sa ajung si in blocurile }
        { vecine plecand din acest bloc   }
    if x>0 then AjungeIn(h,x,y,h,x-1,y,
                  v+d[Offset(h,x-1,y)]);
    if x<w[h]-1 then AjungeIn(h,x,y,h,x+1,
                  y,v+d[Offset(h,x+1,y)]);
    if y>0 then AjungeIn(h,x,y,h,x,y-1,
                  v+d[Offset(h,x,y-1)]);
    if y<w[h]-1 then AjungeIn(h,x,y,h,x,
                  y+1,v+d[Offset(h,x,y+1)])
  end
end;

procedure CalculNivel(h:Integer);
        { Calculeaza lungimile minime ale }
        { drumurilor care ajung la fiecare }
        { bloc dintr-un nivel              }
var i,j:Integer;
begin
  if h=0
  then
  begin { Daca este nivelul baza atunci se }
        { poate intra pe fiecare latura    }
    for i:=0 to n-1 do
    begin { Pentru fiecare bloc de pe      }
    { margine incerc sa construiesc drumul }
    { plecand din acel bloc                }
     AjungeIn(100,0,0,0,i,0,d[Offset(0,i,0)]);
     AjungeIn(100,0,0,0,i,n-1,
                      d[Offset(0,i,n-1)]);
     AjungeIn(100,0,0,0,0,i,d[Offset(0,0,i)]);
     AjungeIn(100,0,0,0,n-1,i,
                      d[Offset(0,n-1,i)])
    end
  end
  else
  begin
{ Daca nu este nivelul baza incerc sa     }
{ ajung la acest nivel prin oricare din   }
{ blocurile existente la nivelul inferior }
    for i:=0 to w[h]-1 do
      for j:=0 to w[h]-1 do
        AjungeIn(h-1,i+1,j+1,h,i,j,
  min[Offset(h-1,i+1,j+1)]+d[Offset(h,i,j)])
  end
end;

procedure AfiseazaDrum;
            { afiseaza drumul minim gasit }
var dh,dx,dy:piramida_byte;{Stocheaza drumul}
    h,x,y,pos,k:Integer;
begin
  k:=0;     { Numar de blocuri prin care a  }
            { trecut drumul                 }
  h:=m-1;  { Pozitia blocului curent h,x,y  }
  x:=0; y:=0;
  while h<100 do
  begin
    dh[k]:=h;       { Stochez acest bloc si }
          { incrementez numarul de blocuri }
    dx[k]:=x; {prin care trece drumul minim}
    dy[k]:=y;
    Inc(k);
    pos:=Offset(h,x,y);
    h:=hback[pos]; {Trec la blocul anterior}
    x:=lback[pos]; y:=cback[pos]
  end;
  Writeln(fout,' ',k);
                { Afisez lungimea drumului }
  for h:=k-1 downto 0 do
            { Afisez in sens invers drumul }
  Writeln(fout,dh[h],' ',dx[h]+1,' ',dy[h]+1)
end;
```

```
Begin
  Assign(fin,'PIRAMIDA.IN');
  Reset(fin);
                          { Fisier de intrare }
  Assign(fout,'PIRAMIDA.OUT');
  Rewrite(fout);
                          { Fisier de iesire }
  Readln(fin,n);  { Citirea lungimii bazei }
  Initializare;
     { Procedura de initializare variabile }
  for k:=0 to m-1 do
    for i:=0 to w[k]-1 do
    begin
      for j:=0 to w[k]-1 do
      Read(fin,d[Offset(k,i,j)]);
                      { Citirea duritatilor }
      Readln(fin)
    end;
  for i:=0 to m-1 do
    CalculNivel(i);
              { Calcularea drumurilor pana }
              { la un fiecare nivel        }
  Write(fout,min[Offset(m-1,0,0)]);
         { Scrierea lungimii minime gasite}
  AfiseazaDrum;
                  { Afisarea drumului gasit }
  Close(fin);
  Close(fout)
End.
```

P079807: Rigle Golomb

Problemă propusă de conf. dr. Henri Luchian, Universitatea "Al. I. Cuza", Iași

În loc de analiza riguroasă a problemei vă vom "istorisi" cele întîmplate la Constanța. "Ideile" olimpicilor pot fi utile ori de câte ori vă veți afla în fața unei probleme având aceleași caracteristici.

Aceasta a fost o problemă la care aproape toți concurenții au reușit să "păcălească" juriul. Problema a fost considerată ca fiind destul de dificilă, comisia așteptându-se la soluții "speciale" din partea concurenților cum ar fi cele bazate pe algoritmii genetici. Nu a fost să fie așa, deoarece comisia a scăpat din vedere faptul că datele de intrare conțineau un singur număr și acesta putea lua valori între 1 și 16. Din această cauză nu existau decât 16 posibilități pentru rezultate, iar fișierul de ieșire nu conținea decât maxim 16 numere. Așa că, spre mai mica sau mai marea surprindere a comisiei, concurenții au declarat niște constante care erau afișate în funcție de valoarea care se găsea in fișierul de intrare. Singurul impediment în obținerea punctajului maxim era determinarea celor 16 soluții posibile. Concurenții au folosit diferite metode pentru a le determina în cele 4 ore pe care le-au avut la dispoziție. S-au folosit algoritmi probabiliști (dar nu genetici), metode backtracking, metode euristice etc. Au rulat programele respective în "background" și au copiat rezultatele în constantele declarate în programele predate pentru evaluare. (Se pare că multi-taskingul din Windows poate fi folosit la ceva!) Și pentru ca participanții să nu se plângă că problema a fost prea dificilă, cea mai mare valoare cu care au fost testate programele a fost 11 (unsprezece). Mai mult, s-a acordat punctaj maxim chiar dacă soluția nu era optimă, ci doar relativ apropiată de ceea ce ar fi trebuit să se obțină, așa că unii concurenți au obținut cele 50 de puncte cu problema rezolvată doar pe jumătate. Mai trebuie precizat faptul că olimpicii nu au ezitat să-și arate "recunoștința" față de generozitatea comisiei și și-au manifestat dorința de a mai rezolva câteva probleme de acest fel. Având în vedere cele precizate mai sus, programul următor ar fi obținut cu siguranță punctajul maxim.

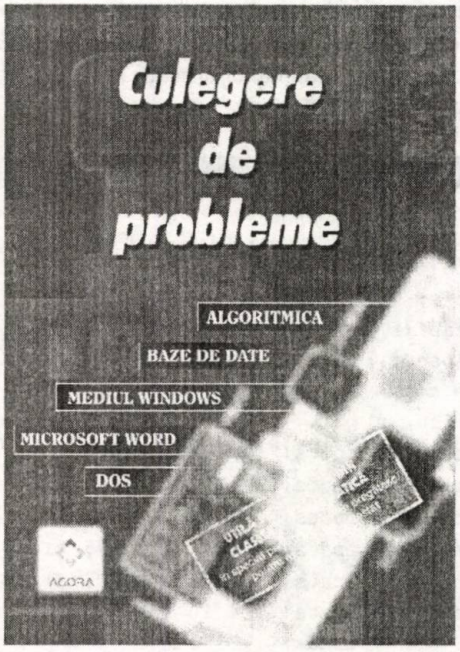
Listing RIGLA.CPP

```
#include <stdio.h>
void main(void){
  FILE *f=fopen("GOL.IN","rt");
  int n;
  fscanf(f,"%d",&n);
  fclose(f);
  f = fopen("gol.out", "wt");
  switch (n){
    case 1: fputs("1", f); break;
    case 2: fputs("1 3", f); break;
    case 3: fputs("1 4 6", f); break;
    case 4: fputs("1 4 9 11", f); break;
    case 5: fputs("1 4 10 15 17", f); break;
    case 6: fputs("1 4 10 18 23 25", f);
            break;
    case 7: fputs("1 4 9 15 22 32 34", f);
            break;
   case 8: fputs("1 5 12 25 27 35 41 44", f);
           break;
    case 9: fputs("1 6 10 23 26 34 41 53 \
55", f); break;
    case 10: fputs("1 4 13 28 33 47 54 64 \
70 72", f); break;
    case 11: fputs("2 6 24 29 40 43 55 68 \
75 76 85", f); break;
    case 12: fputs("2 5 25 37 43 59 70 85 \
89 98 99 106", f); break;
    case 13: fputs("1 6 14 21 23 39 49 73 \
76 105 117 136 147", f); break;
    case 14: fputs("1 4 13 20 28 30 51 65 \
85 108 119 141 175 181", f); break;
    case 15: fputs("1 3 9 19 33 50 55 78 93 \
118 122 157 184 205 218", f); break;
    case 16: fputs("1 3 11 23 36 50 55 79 \
96 117 124 154 188 194 236 251", f); break;
}
fclose(f);
}
```

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Culegerea conține:

- probleme de algoritmică, majoritatea rezolvate în Pascal
- probleme de gestiune a bazelor de date (FOX-PRO)
- probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word
- probleme de verificare a cunoștințelor aferente sistemului de operare DOS

Această culegere constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru atestatul de informatică dar poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, culegerea conține o varietate mare de enunțuri de probleme rezolvate, abordând diferite metode și stiluri de programare.

• • •

Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) mai elegant și cu o eficiență în programare mai ridicată.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422] sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

PROBLEME propuse pentru REZOLVARE

În ultimele două numere ale Ginfo ați fost provocați la întrecere cu problemele de selecție date lotului lărgit de informatică. Din păcate "nu ați ridicat mănușa...", am primit foarte puține rezolvări. În acest număr al revistei supunem atenției problemele cu care s-au confruntat concurenții de la Olimpiada de Informatică a Europei Centrale și de Est.

P089801: Pătrate

Se dau `N` pătrate situate în plan, având laturile paralele cu axele de coordonate. Coordonatele vârfurilor sunt numere întregi. Pătratele nu se ating și nu se suprapun.

Se cere să se determine numărul de pătrate vizibile din originea O, O = (0,0).

Un pătrat este *vizibil* din originea O, dacă există două puncte distincte A și B de coordonate întregi, situate pe o aceeași latură a pătratului, astfel încât interiorul triunghiului OAB nu are puncte comune cu nici unul dintre celelalte pătrate.

Date de intrare

Prima linie a fișierului de intrare **`SQUARES.IN`** conține un număr întreg `N`, `1<=N<=1000`, reprezentând numărul de pătrate.

Fiecare din următoarele `N` linii descrie un pătrat prin numerele întregi `X`, `Y` și `L` separate prin câte un spațiu `1<=X,Y,L<=10000`. `X` și `Y` sunt coordonatele colțului din stânga-jos al pătratului, iar `L` este lungimea laturii.

Date de ieșire

Prima (și singura) linie a fișierului de ieșire **`SQUARES.OUT`** trebuie să conțină numărul pătratelor vizibile din origine.

Exemple

SQUARES.IN	SQUARES.OUT
3 2 6 3 1 4 1 3 4 1	3

SQUARES.IN	SQUARES.OUT
4 1 2 1 3 1 1 2 4 2 3 7 1	2

P089802: Cărți de joc

Alice și Bob au o mulțime de `N` cărți de joc etichetate cu numere de la `1` la `N` (două cărți nu pot avea aceeași etichetă) și o mașină de amestecat cărțile. Presupunem că `N` este număr impar.

Mașina acceptă o mulțime de cărți aranjate arbitrar și realizează următoarea operație de *dublă-amestecare*: pentru fiecare poziție `i`, `1<=i<=N`, în cazul în care cartea de pe poziția `i` are eticheta `j` și cartea de pe poziția `j` are eticheta `k`, atunci după terminarea operației de dublă-amestecare, poziția `i` va fi ocupată de cartea având eticheta `k`.

Alice și Bob se joacă. Alice scrie (pentru ea) numerele de la `1` la `N` într-o ordine oarecare: a_1, ..., a_N. Apoi, rearanjează cărțile astfel încât poziția a_i să fie ocupată de cartea etichetată cu a_{i+1} pentru orice `i`, `1<=i<=N-1`. Poziția a_N va fi ocupată de cartea cu eticheta a_1.

Astfel cărțile sunt așezate în ordinea x_1, x_2, ..., x_N, unde x_i este cartea de pe poziția `i`.

În continuare Alice execută `S` operații de dublă-amestecare, folosind mașina. În final cărțile ajung în ordinea p_1, p_2, ..., p_N pe care Alice o comunică lui Bob. De asemenea îi comunică și numărul `S`. Bob trebuie să găsească ordinea x_1, x_2, ..., x_N în care se aflau inițial cărțile (înainte de prima operație de amestecare).

Date de intrare

Prima linie a fișierului de intrare **CARDS.IN** conține două numere întregi separate printr-un spațiu: numărul întreg impar N, 1<=N<=1000, reprezentând numărul de cărți și numărul întreg S, 1<=S<=1000, reprezentând numărul operațiilor de dublă-amestecare.

Următoarele linii descriu ordinea finală a cărților după executarea celor S operații de dublă-amestecare. Pentru fiecare a i-a operație, 1<=i<=N, linia i+1 din fișierul de intrare conține p_i (eticheta cărții de pe poziția i după efectuarea tuturor operațiilor de dublă-amestecare).

Date de ieșire

Fișierul de ieșire **CARDS.OUT** va conține N linii care descriu ordinea cărților înainte de prima operație de dublă-amestecare.

Pentru orice i, $1 \le i \le N$, linia i din fișierul de ieșire trebuie să conțină o valoare x_i (eticheta cărții de pe poziția i, înainte de prima operație de dublă-amestecare).

Exemple

CARDS.IN	CARDS.OUT
5 2	2
4	5
1	4
5	1
3	3
2	

CARDS.IN	CARDS.OUT
7 4	4
6	7
3	5
1	6
2	1
4	2
7	3
5	

P089803: Reduceri

Fie o secvență de N numere întregi și pozitive: $a=[a_1,a_2,...,a_N]$ asupra căreia se pot efectua operații de reducere.

O *operație de reducere* constă în înlocuirea a două elemente consecutive a_i și a_{i+1} prin diferența a_i-a_{i+1}. Pentru o secvență de N numere întregi, se pot efectua exact N-1 operații de reducere distincte, fiecare generând o nouă secvență cu N-1 elemente.

De exemplu, notăm cu con(a,i) secvența de N-1 elemente obținută din $a=[a_1,a_2,...,a_N]$ prin înlocuirea elementelor a_i și a_{i+1} cu numărul întreg a_i-a_{i+1}: $con(a,i)=[a_1,\ldots,a_{i-1},a_i-a_{i+1},a_{i+2},\ldots,a_N]$

Efectuând N-1 reduceri asupra unei secvențe de N numere întregi, evident se va obține, în final, un singur număr.

De exemplu, efectuând reducerile 2, 3, 2 și 1 în această ordine asupra secvenței [12,10,4,3,5] se obține numărul 4, astfel:

```
con([12,10,4,3,5],2) = [12,6,3,5]
con([12,6,3,5]   ,3) = [12,6,-2]
con([12,6,-2]    ,2) = [12,8]
con([12,8]       ,1) = [4]
```

Dată fiind o secvență $a_1, a_2,..., a_N$ și un număr T, să se determine o succesiune de N-1 reduceri care trebuie efectuate asupra secvenței date inițial, astfel încât să se obțină numărul T.

Date de intrare

Prima linie a fișierului de intrare **SUBTRACT.IN** conține două numere întregi separate printr-un spațiu: N, 1<=N<=100, reprezentând numărul de elemente din secvența inițială, și numărul T, 10000<=T<=10000.

Următoarele N linii conțin elementele secvenței date, fiecare număr a_i pe linie nouă, $1<=a_i<=100$.

Date de ieșire

Fișierul de ieșire **SUBTRACT.OUT** conține N-1 linii, descriind o succesiune de operații de reducere care transformă secvența dată inițial în numărul T. Linia k din fișierul de ieșire conține un număr întreg reprezentând a k-a operație de reducere aplicată secvenței. Se presupune că există cel puțin o asemenea succesiune de reduceri.

Exemple

SUBTRACT.IN	SUBTRACT.OUT
4 5	1
10	2
2	1
5	
2	

SUBTRACT.IN	SUBTRACT.OUT
5 4	2
12	3
10	2
4	1
3	
5	

P089804: Soldați

N soldați din țara *Caroiaj* sunt așezați la întâmplare pe teritoriul țării.

O *poziție* în țara *Caroiaj* este dată de coordona-

tele (x,y), numere întregi. Soldații pot să-și schimbe poziția astfel: într-o mutare, un soldat poate să se deplaseze cu o unitate în sus, în jos, la stânga sau la dreapta (își schimbă fie coordonata x, fie coordonata y cu 1 sau -1).

Soldații vor să se alinieze pe orizontală, unul lângă altul (astfel încât pozițiile lor finale vor fi (x,y), (x+1,y), ..., (x+N-1,y), pentru un anumit x și y). Întregii x și y sunt oarecare. Ordinea finală a soldaților pe linie, este de asemenea oarecare.

Să se determine numărul minim de schimbări de poziție a tuturor soldaților, necesare pentru aliniere.

Doi sau mai mulți soldați nu pot ocupa în același timp aceeași poziție.

Date de intrare

Prima linie a fișierului **SOLDIERS.IN** conține un număr întreg N, 1<=N<=10000, reprezentând numărul de soldați.

Următoarele N linii conțin pozițiile inițiale ale soldaților: pentru orice i, 1<=i<=N, linia i+1 conține o pereche de numere întregi x[i] și y[i] separate printr-un singur spațiu, reprezentând coordonatele soldatului i, -10000<=x[i], y[i]<=10000.

Date de ieșire

Prima și singura linie a fișierului de ieșire **SOLDIERS.OUT** va conține numărul minim total de schimbări de poziție a soldaților, necesar pentru aliniere.

Exemple

SOLDIERS.IN	SOLDIERS.OUT
3 1 0 2 4 3 2	4

SOLDIERS.IN	SOLDIERS.OUT
5 1 2 2 2 1 3 3 -2 3 3	8

P089805: Șosele

N orașe numerotate de la 1 la N sunt legate prin șosele cu sens unic. Fiecărei șosele i se asociază lungimea sa și taxa (exprimată în număr de monede), care trebuie plătită pentru parcurgerea acesteia.

Bob și Alice locuiesc în orașul 1. După ce Bob și-a dat seama că Alice a trișat la jocul cu cărți, s-au certat și a hotărât să se mute în orașul N. El vrea să ajungă acolo cât mai repede posibil, dar stă cam prost cu banii.

Să se determine *cel mai scurt drum* de la orașul 1 la orașul N, care poate fi parcurs cu *cel mult toată suma de bani* pe care-i are Bob la dispoziție.

Date de intrare

Prima linie a fișierului de intrare **ROADS.IN** conține un număr întreg K, 0<=K<=10000, reprezentând numărul maxim de monede pe care le poate cheltui Bob.

A doua linie conține un număr întreg N, reprezentând numărul de orașe, (2<=N<=100).

A treia linie conține un întreg R, 1<=R<=10000, reprezentând numărul de șosele.

Următoarele R linii oferă informații despre fiecare dintre cele R șosele:

- S și D (în această ordine) sunt orașele de la capetele unei șosele, 1<=S, D<=N
- L este lungimea șoselei, 1<=L<=100
- T este taxa de parcurgere a șoselei (exprimată în monede), 0<=T<=100;

S, D, L și T sunt numere întregi separate printr-un singur spațiu.

Rețineți că aceleași două orașe pot fi legate prin șosele diferite.

Date de ieșire

Prima și singura linie din fișierul de ieșire **ROADS.OUT** trebuie să conțină lungimea drumului dintre orașul 1 și orașul N care necesită pentru parcurgere taxe în valoare totală mai mică sau egală decât K.

Dacă nu există drum, în fișierul de ieșire se va scrie numărul -1.

Exemple

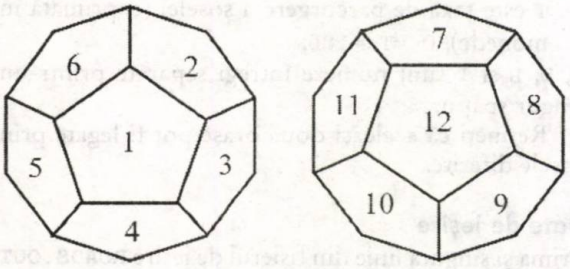
ROADS.IN	ROADS.OUT
5 6 7 1 2 2 3 2 4 3 3 3 4 2 4 1 3 4 1 4 6 2 1 3 5 2 0 5 4 3 2	11

ROADS.IN	ROADS.OUT
0 4 4 1 4 5 2 1 2 1 0 2 3 1 1 3 4 1 0	-1

P089806: Mingea

Profesorul Baltazar este mare microbist. Ziua lui de naștere a fost sărbătorită cu câteva zile înainte de a pleca la Campionatul Mondial de Fotbal '98 din Franța. Prietenii i-au făcut cadou un puzzle în formă de dodecaedru cu care să se joace în timpul meciurilor plictisitoare.

Jucăria are 12 fețe egale, de forma unui pentagon, numerotate de la 1 la 12. Figura de mai jos reprezintă cele două semisfere ale dodecaedrului, cu numerotarea ce va fi utilizată în problemă. Semisferele sunt "lipite" una de alta, astfel încât fața numerotată cu 7 este vecină cu fețele 8, 12, 11, 2 și 6 (două fețe sunt *vecine* dacă au o muchie comună). De exemplu, muchiile *a* și *b* de pe semisfera din stânga, vor fi lipite de muchiile a și b de pe semisfera din dreapta, ca în figura de mai jos.



În plus, sunt date 12 căpăcele în formă de pentagon, numerotate și ele de la 1 la 12. Muchiile căpăcelelor sunt marcate cu numere din mulțimea {0, 1, 2}. Fiecare dintre cel 12 căpăcele poate fi pus pe oricare dintre cele 12 fețe ale dodecaedrului, în orice poziție (dintre cele 5 obținute prin rotirea căpăcelului).

Jocul constă în așezarea căpăcelelor pe fețele dodecaedrului, astfel încât două căpăcele vecine să aibă muchia comună marcată cu același număr.

Ajutați-l pe Profesorul Baltazar să rezolve jocul!

Date de intrare

Fișierul de intrare **BALL.IN** conține 12 linii. Pentru orice `i`, `1<=i<=12`, linia `i` descrie căpăcelul `i` prin specificarea celor cinci numere cu care sunt marcate muchiile sale, separate printr-un spațiu. Această secvență precizează marcajele muchiilor căpăcelului `i`, în ordinea acelor de ceasornic, începând cu o muchie oarecare, (numită în cele ce urmează *muchia de referință* `i`).

Date de ieșire

Fișierul de ieșire **BALL.OUT** va conține descrierea soluției jocului pe 12 linii conținând fiecare două numere întregi. Pentru orice `i`, `1<=i<=12`, linia `i` va conține doi întregi `t[i]` și `n[i]` separați prin câte un singur spațiu, care identifică căpăcelul de pe fața `i` și poziția sa astfel:

Pe fața `i` se pune căpăcelul `t[i]`.

Căpăcelul poate fi pus pe fața `i` în cinci poziții diferite. Poziția exactă va fi specificată prin numărul `n[i]`, care reprezintă numărul feței din dreptul muchiei de referință `t[i]` (privind din centrul feței `i`). Mai precis, muchia de referință a căpăcelului `t[i]` se așează pe dodecaedru astfel încât aceasta se suprapune cu muchia care desparte fața `i` de fața `n[i]`.

Dacă jocul nu are soluție, în fișierul de ieșire va fi scris numai numărul `-1`.

Exemple

BALL.IN	BALL.OUT
0 0 1 1 2	1 2
0 2 1 0 1	3 7
2 0 1 0 1	12 4
0 0 1 2 1	7 9
0 2 1 1 2	9 1
2 0 1 2 1	11 8
0 2 1 2 1	8 2
2 2 1 0 1	4 6
1 2 2 0 0	5 4
0 2 1 0 2	2 12
0 2 1 2 0	6 3
2 0 1 2 0	10 7

BALL.IN	BALL.OUT
1 0 2 0 2	1 2
2 2 2 1 2	2 7
1 1 0 0 0	8 2
1 1 0 2 1	7 1
2 1 1 1 1	11 4
1 2 2 1 1	12 2
2 1 2 2 1	5 2
2 2 0 1 0	3 12
0 1 2 1 2	10 5
2 2 1 0 0	9 3
1 2 0 2 0	6 10
2 2 2 0 1	4 7

Rezultatele echipei României

La CEOI'98 au participat 10 țări.
Echipa României a obținut locul 3, după, echipele Poloniei și Slovaciei.

Mihai Scorțaru (Cluj): argint
Angel Proorocu (Ploiești): argint
Tudor Leu (Focșani): bronz
Diodor Bițan (București): bronz

În dialog cu Mihai Scorțaru

O scurtă privire "ÎNAPOI"

Clara Ionescu

Din interviurile publicate în numerele anterioare ale revistei noastre am avut ocazia să cunoaștem părerea unor profesori universitari despre învățământul liceal de informatică. Înainte de a sta de vorbă cu un profesor de liceu, am hotărât să pun câteva întrebări unui fost elev care a fost "victima" acestei forme de învățământ. De ce am pus ghilimele? Veți afla dacă citiți interviul realizat cu Mihai Scorțaru, absolvent al Liceului de Informatică din Cluj, în prezent student la Universitatea Tehnică din Cluj.

Au trecut doar câteva săptămâni de când nu mai mergi la școală. Povestește-mi, cum e la facultate?

Au trecut de fapt câteva luni de când nu mai trebuie să merg la "școală" și să "suport" diferite "limbi străine". Din acest punct de vedere pot să spun că facultatea a adus o schimbare majoră în sensul că nu ni se mai predau tot felul de lucruri "inutile".

Și de la facultate ar putea fi eliminate unele discipline, dar situația este incomparabil mai bună decât cea din liceu. Această ameliorare a situației are însă și costurile ei, adică a trebuit să mă despart de colegii din liceu care au ales să-și continue studiile în altă parte. Probabil că locul lor (în mintea mea) va fi ocupat de noii colegi, dar nu cred că va fi la fel. Un alt aspect negativ este comportamentul unor proaspeți studenți care erau obișnuiți să obțină rezultate excelente în liceu și de aceea cred că au întotdeauna dreptate și acceptă foarte greu ideile altora; acest fapt nu poate fi în folosul nimănui. Nici profesorii universitari nu renunță prea ușor la părerile lor, chiar dacă se întâmplă ca acestea să fie eronate.

Când te-ai decis să te înscrii la "Poli"? Știm că în funcție de vârstă, toți copiii visează diferite meserii... Niciodată nu ai vrut să fii mecanic de locomotivă, pompier sau cosmonaut?

Nu cred să fi existat vreo perioadă semnificativă de timp în care să fi dorit să urmez altă facultate. Uneori mă gândeam că medicina ar fi fost o alegere mai bună dar am renunțat destul de repede la idee. Referitor la alegerea altor meserii pot să vă spun că am fost o dată pe o locomotivă și chiar am condus câțiva kilometri, dar tot ce mi-am amintit apoi a fost cum se pornește "sirena" locomotivei. Cred că oricărui copil (băiat) i se pare o idee extraordinară să plece în spațiu și de aceea meseria de cosmonaut surâde oricui. Recunosc că aș renunța aproape la orice pentru a putea participa la o misiune spațială, chiar și acum. Nu cred că aș dori să practic vreo meserie care să nu fie strâns legată de informatică... dar nu se știe niciodată.

Atunci când ai terminat clasa a VIII-a, trebuia să alegi un liceu unde să îți continui studiile. Cum te-ai hotărât pentru Liceul de Informatică?

În clasa a VIII-a toată lumea începe să-ți dea sfaturi cu privire la liceul pe care să-l urmezi. Aproape toți profesorii îți recomandă un liceu care să aibă profilul apropiat de materia pe care o predau ei. Așa că, dacă aș fi luat în considerare toate sfaturile, mi-ar fi fost foarte greu să aleg între matematică, biologie, limbi străine etc. Observați că nici un profesor nu s-a gândit să-mi recomande informatica, deoarece nici unul nu cunoștea aproape deloc domeniul. Cum nici eu nu cunoșteam aproape nimic despre informatică, decizia de a urma acest liceu a venit pur și simplu din întâmplare. Îmi plăceau foarte mult jocurile mecanice și am auzit că pe la liceele cu clase de informatică elevii se mai și jucau. După ce am hotărât că trebuie să urmez o clasă de informatică au apărut alte recomandări ale profesorilor: unul avea un fiu la un liceu, altul cunoștea pe cineva altundeva. Până la urmă "candidații" cei mai importanți au rămas Liceul de Informatică și Liceul "Avram Iancu". Părerile "sfătuitorilor" erau împărțite; mai mult, părinții mei au încercat și ei să mă influențeze, opțiunea lor îndreptându-se spre Liceul "Avram Iancu" datorită

distanței mai scurte pe care ar fi trebuit să o parcurg până la școală. Argumentul decisiv a venit din partea doamnei diriginte care avea un fiu la Liceul "Avram Iancu" și care mi-a prezentat situația din acel liceu într-un mod care m-a făcut să renunț la ideea de a merge acolo. Acest lucru, împreună cu dorința (aproape) normală de a *nu face ce vor părinții*, m-a determinat să aleg Liceul de Informatică și cred că am făcut cea mai bună alegere care se putea face în situația actuală din România.

Acum, după ce s-a încheiat capitolul numit "liceu", ce părere ai despre acești ani de școală? A fost ușor? A fost greu? Ce ți-a plăcut? Ce nu?

Se spune că anii de liceu sunt cei mai frumoși ani din viața unui om și cum eu încă nu am trecut prin facultate, nu pot decât să fiu de acord cu această afirmație. Nu se poate spune că e ușor la Liceul de Informatică, ca dovadă mai mulți dintre colegii mei nu au rezistat patru ani. Dificultatea apare din cauză că toți profesorii (cu mici excepții) cred că materia lor este esențială pentru pregătirea unui informatician care în opinia lor trebuie neapărat să știe și care este capitala Coastei de Fildeș și care sunt nervii cranieni și ce înseamnă izopropilbenzen și exemplele ar putea continua. Au existat și părți bune în liceu cum ar fi atmosfera generală, unele materii interesante predate de profesori foarte bine pregătiți și nu în ultimul rând: jocurile pe calculator.

Dacă - printr-o minune - ar trebui s-o iei de la capăt, la ce liceu te-ai înscrie?

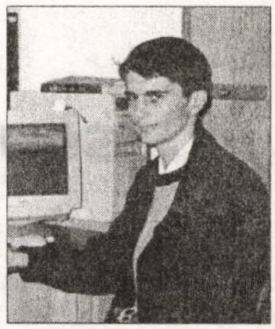
Nu pot să vă răspund la o astfel de întrebare. Încă nu există un liceu în România care să îndeplinească condițiile necesare pentru ca un elev să dorească să se înscrie acolo. Poate că în viitorul mai mult sau mai puțin îndepărtat...

Presupunând că îți cere sfatul un copil de clasa a VIII-a, i-ai recomanda un liceu sau o clasă de informatică? Ce ar câștiga (sau pierde) el/ea dacă ar urma drumul traversat de tine? Rezultă că te întreb ce ai câștigat (sau pierdut) în anii de liceu?

Nu aș recomanda un liceu sau o clasă de informatică decât unui copil care știe sigur că vrea să facă programare. Multă lume crede, pe bună dreptate, că într-un astfel de liceu elevii vor învăța numeroase lucruri interesante despre calculatoare, dar în realitate, în aceste licee se pune un *prea mare accent pe programare*. Pot să-i spun unui copil că dacă ar urma exact drumul traversat de mine ar reuși să obțină o oarecare recunoaștere a rezultatelor și efortului depus, ar avea un bogat bagaj de cunoștințe în domeniul programării dar ar pierde numeroase evenimente importante din viața unui licean din cauza faptului că nu se va putea afla întotdeauna printre colegi. Ar mai putea obține destul de ușor, cred eu, și admiterea fără examen în învățământul superior.

Ce consideri că ai învățat inutil la școală? Dacă nu ar fi existat Secția

Carte de vizită



Mihai Scorțaru este student în anul întâi la Universitatea Tehnică din Cluj, Facultatea de Automatică și Calculatoare, secția Calculatoare cu predare în limba engleză. A absolvit Liceul de Informatică "Tiberiu Popoviciu" din Cluj.

În perioada 1996-1998 a participat la numeroase concursuri de programare, cele mai importante rezultate ale sale fiind:

- **mențiune la Olimpiada Națională de Informatică, 1996**
- **premiul II la Olimpiada Națională de Informatică, 1997**
- **mențiune la Olimpiada Națională de Informatică, 1998**

Concursuri internaționale:

- **medalie de argint la Olipiada Balcanică de Informatică din Drama, Grecia 1997**
- **medalie de argint la Olimpiada de Informatică a Europei Centrale și de Est din Zadar, Croația 1998**
- **medalie de argint la Olimpioada Internațională de Informatică din Setúbal, Portugalia 1998**

În 1996 și 1998 a participat la finala concursului "Nemes Tihamér" din Budapesta, concurs de informatică la care sunt selecționați componenții echipei Ungariei pentru diferite concursuri internaționale.

Începând din 1997 este colaborator al *GInfo*, fiind autorul mai multor articole și prezentând soluții la numeroase probleme propuse.

În prezent este angajat ca programator la firma S.C. ELECTRIC 21 SERVICE S.R.L. din Cluj-Napoca, firmă ce are ca domeniu de activitate proiectarea asistată de calculator a instalațiilor electrice și termice.

Calculatoare în învățământul superior, ar fi fost posibil să te înscrii la Medicină sau la Filosofie?

Lucrurile inutile pe care le-am învățat la școală sunt atât de multe încât nu cred că enumerarea lor ar încăpea în paginile acestei reviste. Pe scurt acestea au fost: biologie, istorie, geografie, logică, psihologie, filosofie, chimie și chiar o parte din literatura română. Mulți se vor arăta scandalizați față de această ultimă afirmație, dar majoritatea elevilor consideră, pe bună dreptate, că este inutil studiul atât de detaliat al acestor discipline, așa cum se face în liceu. Când aminteam cuiva acest lucru ni se spunea că trebuie să învățăm să vorbim corect românește. Din câte probabil v-ați dat seama, eu știu să vorbesc destul de bine românește și vă mărturisesc că destul de des "am cam trecut pe lângă" literatură în liceu. Ceea ce mă miră și mai mult este faptul că, atunci când se discuta eventualitatea eliminării unor materii din planul de învățământ, ni se spunea că nu se poate, deoarece profesorul respectiv ar rămâne fără loc de muncă. Ceea ce înțelegeam eu, și nu mi-am schimbat părerea, era că pentru binele unui singur om erau sacrificați peste 100 de elevi care trebuia să învețe o serie de lucruri pe care le preda acel profesor. M-aș fi putut înscrie la Medicină, așa cum am mai spus, dar niciodată nu am avut înclinații spre filozofie.

Dacă ar sta în puterea ta, ce ai schimba în planul de învățământ, respectiv în programele școlare actuale, pe care le cunoști foarte bine, ținând cont de faptul că patru ani ai învățat pe baza lor?

Nu cred că vreo modificare adusă actualului plan de învățământ ar aduce îmbunătățiri semnificative. După părerea mea ar trebui creat de la zero un nou plan de învățământ, la elaborarea căruia să fie consultați și elevii. Ar trebui micșorat numărul de ore pe care un elev trebuie să le petreacă la școală, numărul de materii studiate și cantitatea de cunoștințe pe care elevul trebuie să și le însușească. O descriere a tuturor programelor școlare actuale nu cred că ar fi oportună, de aceea mă voi limita la unele precizări referitoare la programa de informatică, programă care cred că vă interesează pe dumneavoastră, având în vedere profilul acestei reviste. În opinia mea, nu ar trebui să se pună semnul de egalitate între *informatică*, luată ca știință, și *programare*. Dintre cei care frecventează o clasă cu profil de informatică, foarte puțini vor ajunge programatori deoarece, chiar dacă la ora actuală meseria de programator este una dintre cele mai bine plătite, locurile de muncă în acest domeniu sunt limitate. Ar trebui să se pună accent mai mare pe utilizarea calculatorului și a celor mai importante pachete software. Mai mult, ar trebui ca elevii să fie familiarizați cu Internet-ul; în timpul celor 4 ani pe care i-am petrecut în liceu nu cred că au fost acordate mai mult de 5 ore acestui subiect. Cred că am colegi care nu știu ce înseamnă o pagină Web sau care nu au auzit niciodată de Internet Explorer sau Netscape Navigator. Mai mult, cunosc mulți elevi foarte bine pregătiți în domeniul programării, dar care în afară de limbajul Pascal nu cunosc nimic altceva. Pot să afirm că limbajul Pascal nu va folosi nimănui în găsirea unui loc de muncă, deoarece acum se folosesc foarte mult tehnicile de programare vizuală, respectiv limbajele Visual Basic, Visual C++ etc. și se încearcă producerea de software care să poată rula pe orice platformă (Java). Nu cred că în timpul liceului și nici chiar în facultate un elev sau un student va învăța așa ceva pentru simplul motiv că nu există suficienți profesori care să cunoască acest lucru și nu există dotarea necesară. Există numeroase posturi care cer cunoașterea anumitor pachete software și cred că spre acest domeniu ar trebui să se îndrepte cei care predau informatica în liceu. Pentru cei care doresc să învețe programare ar trebui să existe clase sau licee speciale, dar nu ar trebui ca toți elevii de la clasele de informatică să știe programare doar pentru că se spune că "numai așa informaticienii români pot obține rezultate excepționale la concursurile internaționale de informatică". Nu cred că societatea românească ar avea de câștigat de pe urma celor 4 - 5 elevi care obțin medalii la aceste concursuri, elevi care cu mici excepții, vor părăsi țara cu prima ocazie.

Sunt oare lucruri pe care le-ai învățat la școală și acum, de când ești student ar "trebui" să le uiți?

După părerea profesorului meu de programare ar trebui să uit tot ceea ce nu ține de programarea structurată pentru că ... așa vrea dânsul (altă explicație nu am). Altceva nu cred că ar trebui să uit, dar observ că unii profesori consideră că ar trebui să știu unele lucruri de care nu am auzit niciodată...

Consideri că ai fi putut învăța informatica singur?

În nici un caz. Eu în clasa a IX-a nu știam (aproape) nimic în domeniul programării. Cunoșteam limbajul Basic dar aceast lucru nu mi-a folosit la nimic. Dacă nu aș fi avut norocul să am profesori foarte bine pregătiți nu cred că aș fi reușit să aflu cam ce "vrea programarea de la mine". Aș vrea să profit de această ocazie pentru a mulțumi tuturor celor care s-au ocupat de pregătirea mea. Unii spun că un bun programator este (a fost) un autodidact, că nimeni nu este în stare să-l pregătească. Îmi cer scuze dacă supăr pe cineva, dar trebuie să spun că eu nu am învățat singur nici un algoritm sau metodă de programare, ci cu ajutorul profesorilor din liceu sau cu ocazia taberelor de antrenament organizate în scopul pregătirii lotului național de informatică în ultimii 2 ani. Nu spun că nu aș fi putut învăța și singur aceste lucruri, dar este mult mai ușor dacă te ajută cineva care are experiență în acest domeniu. Probabil că atunci când nu găsești pe nimeni care să te ajute, recurgi și la cărți, așa cum am făcut și eu când am învățat Java, Html, Visual Basic și altele.

Mulțumesc și succes !

TI & C
în reforma curriculară

Anca Voicu

De-a lungul istoriei societatea a trecut prin trei etape mari: societatea agricolă, societatea industrială și societatea informațională (Toffler). Evident, societatea de azi poate fi considerată ca fiind "network society" adică o societate în care accesul la informație se realizează la scară mondială prin mijloace moderne de comunicație.

Transformările societății românești din ultimii ani, dezvoltarea și răspândirea informaticii, pătrunderea hardware-ului și software-ului modern în țara noastră, impun o pregătire diversificată a tinerilor în acest domeniu. Învățământul preuniversitar trebuie să fie preocupat ca elevii să dobândească toate cunoștințele necesare în utilizarea tehnologiei informaționale și comunicaționale.

Proiectul Ministerului Educației Naționale

Realizând importanța pregătirii tinerilor "pentru mâine", Ministerul Educației Naționale a introdus în curriculumul național (planul de învățământ) disciplina **"Tehnologia Informației"** *ca disciplină de sine stătătoare dar și sub formă interdisciplinară.*

Obiective

Înainte de toate trebuia clarificat ce anume va urmări această disciplină, ce obiective educative și instructive se vor atinge datorită ei?

Consider că toți cititorii acestui articol vor fi de acord că acest învățământ "reformat" trebuie *să dezvolte capacitățile tinerilor privind utilizarea resurselor de tehnologie informațională și comunicațională.* Dar nu este cazul unei discipline izolate. Însăși rolul ei, motivele introducerii în planul de învățământ, exclud acest lucru. Elevii vor trebui să învețe *să folosească tehnologia informațională și comunicațională ca suport în învățarea altor discipline.*

Pornind de la faptul că nu există domeniu de activitate unde să nu se prelucreze și să nu se transmită informații atât în cadrul domeniului respectiv cât și spre exteriorul lui, toți membrii societății ("de la mic la mare") trebuie să înțeleagă că azi *informația* este o *valoare*. Copiii trebuie să se obișnuiască cât de repede se poate *să folosească tehnologia informațională și comunicațională ca instrument de accesare, procesare, prezentare și transmitere a informației.*

Totodată ***elevii trebuie să conștientizeze influența informaticii asupra societății și invers.*** Dezvoltarea impetuoasă a tehnologiei informaționale și comunicaționale modifică societatea, de aici rezultând necesitatea înțelegerii rolului pe care îl au calculatoarele în schimbările din viața socială, economică și ale aspectelor etice în ceea ce privește utilizarea lor.

Elevii trebuie *să înțeleagă implicațiile tehnologiei informaționale și comunicaționale într-o societate modernă.*

Deprinderi

Vorbind mai la concret, prin disciplina "Tehnologia Informației" se urmărește **dezvoltarea deprinderilor elevilor, în domeniile:**

- ***comunicarea informației*** *- utilizarea TIC pentru a obține, prezenta și comunica informația în forme variate: text, grafică, tabele, histograme, sunet etc.*

  (Procesoare de text, Procesoare grafice, Desk Top Publishing, e-mail, Internet, video conferințe)
- ***mânuirea datelor (informațiilor)*** *- selectarea și sortarea informațiilor de la surse de date organizate, crearea și utilizarea bazelor de date. Pre-*

Listing ESCHIMOSI.PAS

```
{$N+}
Program Eschimos;
var n,i:Byte;
    a,b,c:Comp;
    fi,fo:Text;
Begin
  Assign(fi,'ESC.IN'); Reset(fi);
  Readln(fi,n); Close(fi);
  Assign(fo,'ESC.OUT'); Rewrite(fo);
  a:=0; b:=1; c:=a+b;
  for i:=2 to n do
  begin
    a:=b; b:=c; c:=a+b;
    a:=b; b:=c; c:=a+b
  end;
  Writeln(fo,c:15:0); Close(fo)
End.
```

P079804: Rezistențe

Problemă propusă și rezolvată de Valentin Gheorghiță, student, Universitatea Politehnică, București

Primul lucru delicat în rezolvarea acestei probleme este structurarea datelor. Pentru a putea păstra informațiile despre fiecare rezistență vom crea o listă cu N (numărul de rezistențe la un moment dat) înregistrări, pentru fiecare rezistență păstrând nodurile i și j (i<j) între care se află rezistența și valoarea rezistenței.

În continuare vom crea cele trei proceduri ce tratează transformările prezentate (serie, paralel, triunghi-stea) în problemă. Totdeauna, în momentul apelării uneia dintre aceste proceduri lista cu rezistențe va fi sortată crescător în funcție de i (nodul cu valoarea cea mai mică de care este legată rezistența) iar în caz de egalitate în funcție de j (nodul cu valoarea cea mai mare de care este legată rezistența). Sortarea se va face utilizând algoritmul qsort, complexitatea lui medie fiind $n*\log_2 n$.

Procedura de legare în paralel are complexitatea liniară, egală cu numărul de rezistențe aflate în circuit. În procedură se va parcurge lista cu rezistențe, iar dacă două sau mai multe rezistențe se află între aceleași două noduri, atunci acele rezistențe vor fi eliminate din listă și în locul lor se va adăuga o singură rezistență aflată între aceleași noduri, dar cu valoare calculată după formula dată în problemă.

Pentru transformarea triunghi-stea se va defini gradul unui nod ca fiind numărul de rezistențe care sunt legate de acel nod. În continuare, în procedura care rezolvă această transformare, pentru fiecare două rezistențe aflate între nodurile i și j, respectiv i și k (adică au un nod comun), dacă nodurile i, j, k au gradul mai mare decât 2 se va verifica dacă există o rezistență între nodurile j și k. Dacă există, atunci se vor elimina cele trei rezistențe, introducându-se în locul lor alte trei cu valori calculate după formulele date, legate la un capăt de un nod nou introdus, iar la celălalt de nodurile i, j, respectiv k.

Pentru legarea în serie, se observă că, dacă un nod are gradul doi și nu este unul dintre nodurile între care trebuie calculată rezistența echivalentă, atunci el poate fi eliminat împreună cu rezistențele care se leagă de el. În cazul găsirii unui astfel de nod se parcurge recursiv circuitul într-o parte și în cealaltă, până se va ajunge la noduri care au gradul diferit de 2. Rezistențele și nodurile (cu gradul egal cu 2) prin care s-a trecut în această parcurgere, sunt eliminate, introducându-se o nouă rezistență care are ca valoare suma rezistențelor eliminate și este legată de cele două noduri în care ne-am oprit cu parcurgerea.

În final se vor utiliza procedurile definite pentru a calcula rezistența echivalentă. Pentru aceasta, vom repeta următoarea secvență până când lista rezistențelor va conține o singură înregistrare care este chiar rezistența echivalentă ce trebuia calculată:

- sortare
- procedura legare în paralel
- sortare
- procedura transformare triunghi-stea
- sortare
- procedura legare în serie

Observație: rezultatul se cere cu trei zecimale exacte, deci trebuie puțină atenție la afișarea cu autoformatare, pentru că de exemplu, în Pascal, aceasta produce rotunjirea și nu trunchierea rezultatului.

Listing: REZISTENTE.PAS

```
{$M 65000,0,655360}
program rezistenta_echivalenta;
const ni='REZ.IN';
      no='REZ.OUT';
type sir=array[1..10000] of Integer;
     psir=^sir;
     vect=array[1..10000] of Real;
     pvect=^vect;
var grad,start,fin,a,b,aa,bb,s,v1,v2:psir;
    v, vv:pvect;
    f:Text;
    n,m,nm,i,j,x,y,pz,med,li,ls:Integer;
    aux,nod,nnod,ad,capat,inceput:Integer;
    sfarsit,p,sursa,dest:Integer;
    r,suma,k,auxr:Real;

procedure load;
begin
  New(grad); New(start);
  New(fin); New(s);
  New(a); New(b);
  New(v); New(aa);
  New(bb); New(vv);
  New(v1); New(v2);
  Assign(f,ni); Reset(f);
  Readln(f,n);
  Readln(f,sursa,dest);
  m:=0;
```

```
  while not Seekeof(f) do
  begin
    Readln(f,i,j,k);
    if i>j
    then
    begin
      aux:=i; i:=j; j:=aux
    end;
    m:=m+1;
    a^[m]:=i;
    b^[m]:=j;
    v^[m]:=k
  end;
  Close(f);
  m:=m+1;
  a^[m]:=sursa;
  b^[m]:=n+1;
  v^[m]:=0;
  m:=m+1;
  a^[m]:=dest;
  b^[m]:=n+2;
  v^[m]:=0;
  n:=n+2
end;

procedure qsort(li,ls:Integer);
var i,med:Integer;
begin
  i:=li;
  j:=ls;
  med:=a^[(li+ls) shr 1];
  repeat
    while a^[i]<med do i:=i+1;
    while a^[j]>med do j:=j-1;
    if i<=j
    then
    begin
      aux:=a^[i]; a^[i]:=a^[j]; a^[j]:=aux;
      aux:=b^[i]; b^[i]:=b^[j]; b^[j]:=aux;
      auxr:=v^[i]; v^[i]:=v^[j]; v^[j]:=auxr;
      i:=i+1;
      j:=j-1
    end
  until i>j;
  if li<j
  then qsort(li,j);
  if i<ls
  then qsort(i,ls)
end;

procedure qqsort(li,ls:Integer);
var i,j,med:Integer;
begin
  i:=li;
  j:=ls;
  med:=b^[(li+ls) shr 1];
  repeat
    while b^[i]<med do i:=i+1;
    while b^[j]>med do j:=j-1;
    if i<=j
    then
    begin
      aux:=a^[i]; a^[i]:=a^[j]; a^[j]:=aux;
      aux:=b^[i]; b^[i]:=b^[j]; b^[j]:=aux;
      auxr:=v^[i]; v^[i]:=v^[j]; v^[j]:=auxr;
      i:=i+1;
      j:=j-1
    end
  until i>j;
  if li<j
  then qqsort(li,j);
  if i<ls
  then qqsort(i,ls)
end;

procedure sort;
begin
  qsort(1,m);
  x:=1;
  while x<=m do
  begin
    y:=x;
    while (a^[y]=a^[x]) and (y<=m) do y:=y+1;
    qqsort(x,y-1);
    x:=y
  end
end;

procedure paralel;
begin
  nm:=0;
  x:=1;
  while x<=m do
  begin
    y:=x;
    while (a^[y]=a^[x]) and (b^[y]=b^[x])
                                and (y<=m) do
      y:=y+1;
    if y=x+1
    then r:=v^[x]
    else
      begin
        r:=0;
        for i:=x to y-1 do r:=r+1/v^[i];
        r:=1/r
      end;
    nm:=nm+1;
    aa^[nm]:=a^[x];
    bb^[nm]:=b^[x];
    vv^[nm]:=r;
    x:=y
  end;
```

```
  m:=nm;
  for i:=1 to m do
  begin
    a^[i]:=aa^[i];
    b^[i]:=bb^[i];
    v^[i]:=vv^[i]
  end
end;

procedure det;
begin
  for i:=1 to n do
  begin
    start^[i]:=0;
    fin^[i]:=0
  end;
  x:=1;
  while x<=m do
  begin
    y:=x;
    while (a^[y]=a^[x]) and (y<=m) do
      y:=y+1;
    nod:=a^[x];
    start^[nod]:=x;
    fin^[nod]:=y-1;
    x:=y
  end;
  for i:=1 to n do
    grad^[i]:=0;
  for i:=1 to m do
  begin
    nod:=a^[i];
    nnod:=b^[i];
    grad^[nod]:=grad^[nod]+1;
    grad^[nnod]:=grad^[nnod]+1
  end
end;

procedure cauta(li,ls:Integer);
begin
  if li<=ls
  then
  begin
    med:=(li+ls) shr 1;
    if nnod=b^[med]
    then pz:=med
    else
      if nnod<b^[med]
      then cauta(li,med-1)
      else cauta(med+1,ls)
  end
end;

procedure tr_stea;
begin
 for i:=1 to m do s^[i]:=0;
ad:=0;
nm:=0;
for i:=1 to n do
 if grad^[i]>2
 then
   if not((start^[i]=0) and (fin^[i]=0))
   then
     for x:=start^[i] to fin^[i]-1 do
       if s^[x]=0
       then
         for y:=x+1 to fin^[i] do
           if s^[y]=0
           then
           begin
             nod:=b^[x];
             nnod:=b^[y];
             if (grad^[nod]>2) and
                              (grad^[nnod]>2)
             then
             if not((start^[nod]=0) and
                             (fin^[nnod]=0))
             then
             begin
               li:=start^[nod];
               ls:=fin^[nod];
               pz:=0;
               cauta(li,ls);
               if pz<>0
               then
                if s^[pz]=0
                then
                begin
                  suma:=v^[x]+v^[y]+v^[pz];
                  ad:=ad+1;
                  nm:=nm+1;
                  aa^[nm]:=i;
                  bb^[nm]:=n+ad;
                vv^[nm]:=(v^[x]*v^[y])/suma;
                  nm:=nm+1;
                  aa^[nm]:=nod;
                  bb^[nm]:=n+ad;
                vv^[nm]:=(v^[x]*v^[pz])/suma;
                  nm:=nm+1;
                  aa^[nm]:=nnod;
                  bb^[nm]:=n+ad;
                vv^[nm]:=(v^[y]*v^[pz])/suma;
                  s^[x]:=1;
                  s^[y]:=1;
                  s^[pz]:=1;
                  y:=fin^[i]
                end
             end
           end;
n:=n+ad;
for i:=1 to m do
 if s^[i]=0
```

```
  then
  begin
    nm:=nm+1;
    aa^[nm]:=a^[i];
    bb^[nm]:=b^[i];
    vv^[nm]:=v^[i]
  end;
 for i:=1 to m do
 begin
  a^[i]:=aa^[i];
  b^[i]:=bb^[i];
  v^[i]:=vv^[i]
 end
end;

procedure rec(nod,tata:Integer);
var ch:Boolean;
begin
  nnod:=tata;
  ch:=false;
  if nod>nnod
  then
  begin
    aux:=nod; nod:=nnod; nnod:=aux;
    ch:=true
  end;
  li:=start^[nod];
  ls:=fin^[nod];
  cauta(li,ls);
  s^[pz]:=1;
  suma:=suma+v^[pz];
  if ch
  then
  begin
    aux:=nod; nod:=nnod; nnod:=aux
  end;
  if grad^[nod]=2
  then
    if v1^[nod]=tata
    then rec(v2^[nod],nod)
    else rec(v1^[nod],nod)
  else capat:=nod
end;

procedure serie;
begin
  for i:=1 to n do
  begin v1^[i]:=0; v2^[i]:=0 end;
  for i:=1 to m do
  begin
    nod:=a^[i]; nnod:=b^[i];
    if grad^[nod]=2
    then
      if v1^[nod]=0
      then v1^[nod]:=nnod
      else v2^[nod]:=nnod;
    if grad^[nnod]=2
    then
      if v1^[nnod]=0 then v1^[nnod]:=nod
                     else v2^[nnod]:=nod
  end;
  for i:=1 to m do
    s^[i]:=0;
  nm:=0;
  for i:=1 to m do
    if s^[i]=0
    then
    begin
      nod:=a^[i]; nnod:=b^[i]; p:=0;
      if grad^[nod]=2 then p:=nod;
      if grad^[nnod]=2 then p:=nnod;
      if p<>0
      then
      begin
        suma:=0;
        rec(v1^[p],p);
        inceput:=capat;
        rec(v2^[p],p);
        sfarsit:=capat;
        nm:=nm+1;
        if inceput>sfarsit
        then
        begin
          aux:=inceput;
          inceput:=sfarsit;
          sfarsit:=aux
        end;
        aa^[nm]:=inceput; bb^[nm]:=sfarsit;
        vv^[nm]:=suma
      end
    end;
    for i:=1 to m do
      if s^[i]=0
      then
      begin
        nm:=nm+1;
        aa^[nm]:=a^[i]; bb^[nm]:=b^[i];
        vv^[nm]:=v^[i]
      end;
    m:=nm;
    for i:=1 to m do
    begin
      a^[i]:=aa^[i]; b^[i]:=bb^[i];
      v^[i]:=vv^[i]
    end
end;
procedure scrie;
begin
  Assign(f,no); Rewrite(f);
  Writeln(f,Trunc(v^[1]*1000)/1000:0:3);
  Close(f)
end;
```

```
Begin
  load;
  while m>1 do
  begin
    sort;
    paralel;
    det;
    tr_stea;
    sort;
    det;
    serie
  end;
  scrie
End.
```

P079805: Seiful

Problemă propusă și rezolvată de prof. Dana Lica, Liceul "I. L. Caragiale", Ploiești

Pentru orice grup care trebuie format din `n-2` persone (conform cerințelor problemei) trebuie să existe cel puțin un lacăt pe care nu-l va deschide nimeni din grup. Rezultă că celelalte două persoane rămase în afara grupului dețin fiecare cheia lacătului respectiv. Astfel, atunci când una din cele două persoane intră în grupul de `n-2` persoane, lacătul va putea fi deschis.

Deducem că la fiecare lacăt trebuie să existe exact două chei. Tot de aici se deduce că numărul de lacăte este egal cu numărul grupurilor distincte formate din `n-2` persoane, adică `n(n-1)/2`.

Atunci numărul total de chei va fi `n(n-1)`.

Listing: LACATE.PAS

```
program Chei_si_Lacate;
var n:Longint;

procedure citeste;
var f:Text;
begin
  Assign(f,'SEIF.IN'); Reset(f);
  Readln(f,n); Close(f)
end;

procedure scrie;
var f:Text; i,j:Longint;
begin
  Assign(f,'SEIF.OUT'); Rewrite(f);
  Writeln(f,n*(n-1) shr 1,' ',n*(n-1));
  for i:=1 to n-1 do
    for j:=i+1 to n do Writeln(f,i,' ',j);
  Close(f)
end;

Begin
  citeste;
  scrie
End.
```

P079806: Piramida lui Keops

Problemă propusă și rezolvată de Marius Vlad, student, Universitatea Politehnică, București

Deoarece problema cere determinarea unui *minim* care ar trebui căutat în mulțimea unor soluții posibile care se pot genera arborescent, s-a ales ca metodă de rezolvare metoda programării dinamice.

Vom calcula cu ajutorul unei proceduri recursive (`AjungeIn`) drumurile minime de la fiecare bloc de pe o latură exterioară a bazei piramidei la celelalte blocuri ale bazei.

Pentru fiecare nivel (diferit de bază) și pentru fiecare bloc în parte de pe acest nivel, presupunem că ajungem în acel bloc venind de pe nivelul inferior și calculăm cu ajutorul aceleiași proceduri recursive lungimea drumului plecând de la blocul respectiv la toate blocurile existente pe acel nivel.

Procedura recursivă `AjungeIn` este asemănătoare cu o procedură *fill* obișnuită și poate fi rapid implementată în timp de concurs. Se oprește în momentul în care lungimea drumului pe calea urmată este mai mare decât lungimea drumului minim găsit.

De fiecare dată când se găsește un drum de lungime mai mică decât cel existent se salvează adresa blocului precedent pentru a reconstrui drumul de întoarcere plecând de la blocul din vârf.

Listing PIRAMIDA.PAS

```
{$M 65500,0,655360}
program Piramida;
const DIMAX=4500; {Lungime maxima a
   vectorului in care se salveaza piramida }
     NMAX=30;
       { Lungimea maxima a bazei piramidei }
type piramida_byte=array[0..DIMAX] of Byte;
     piramida_long=array[0..DIMAX] of Longint;

var i,j,k,n,m:Integer;
    d,              { Duritatile blocurilor }
    hback,lback,cback:piramida_byte;
                { Adresa blocului precedent }
    min:piramida_long;
       { Lungimea minima pana la acel bloc }
    pos:array[0..NMAX] of Longint;
    w:array[0..NMAX] of Integer;
    fin,fout:Text;

procedure Initializare;
{ Initializeaza variabile, precalculeaza date }
var s,k:Longint;
begin
  m:=(n div 2)+1;  { Inaltimea piramidei    }
  w[0]:=n;         { Dimensiunea bazei      }
  pos[0]:=0;       { Pozitia de inceput in }
                   { vector a bazei         }
  s:=0;
```

```
  for k:=1 to m-1 do
  begin
    w[k]:=w[k-1]-2;{Dimensiunea nivelului k }
    s:=s+Longint(w[k-1]*w[k-1]);
    pos[k]:=s          { Pozitia de inceput in }
                       { vector a nivelului k  }
  end;
  for k:=0 to DIMAX-1 do min[k]:=MaxLongInt
     { Initializare vector lungimi gasite }
end;

function Offset(h,l,c:Integer):Longint;
    { Afla pozitia in vector a blocului de }
    { piramida de la nivelul h, linia l si }
    { coloana c                            }
begin
  Offset:=Longint(pos[h])+Longint(l*w[h]+c)
end;

procedure AjungeIn(bh,bx,by,h,x,y:Integer;
                                  v:Longint);
var pos:Integer;
{ bh,bx,by - Adresa blocului vecin de unde }
                                { s-a plecat }
{ h,x,y - Adresa blocului din care se refac}
                                { drumurile }
{ v - Lungimea drumului pana la acel bloc
    inclusiv }
begin
  pos:=Offset(h,x,y);
  if v<min[pos]
  then
  begin   { Daca drumul gasit este mai mic }
    min[pos]:=v;{ Marchez lungimea drumului }
    hback[pos]:=bh;
    { Si adresa blocului din care am venit }
    lback[pos]:=bx;
{ pentru a gasi drumul minim plecand de la }
    cback[pos]:=by;          { blocul de sus }
        { Incerc sa ajung si in blocurile   }
        { vecine plecand din acest bloc     }
    if x>0 then AjungeIn(h,x,y,h,x-1,y,
                  v+d[Offset(h,x-1,y)]);
    if x<w[h]-1 then AjungeIn(h,x,y,h,x+1,
                  y,v+d[Offset(h,x+1,y)]);
    if y>0 then AjungeIn(h,x,y,h,x,y-1,
                  v+d[Offset(h,x,y-1)]);
    if y<w[h]-1 then AjungeIn(h,x,y,h,x,
                  y+1,v+d[Offset(h,x,y+1)])
  end
end;

procedure CalculNivel(h:Integer);
        { Calculeaza lungimile minime ale  }
        { drumurilor care ajung la fiecare }
        { bloc dintr-un nivel              }
var i,j:Integer;
begin
  if h=0
  then
  begin { Daca este nivelul baza atunci se }
        { poate intra pe fiecare latura    }
    for i:=0 to n-1 do
    begin { Pentru fiecare bloc de pe      }
    { margine incerc sa construiesc drumul }
    { plecand din acel bloc                }
     AjungeIn(100,0,0,0,i,0,d[Offset(0,i,0)]);
     AjungeIn(100,0,0,0,i,n-1,
                        d[Offset(0,i,n-1)]);
     AjungeIn(100,0,0,0,0,i,d[Offset(0,0,i)]);
     AjungeIn(100,0,0,0,n-1,i,
                        d[Offset(0,n-1,i)])
    end
  end
  else
  begin
 { Daca nu este nivelul baza incerc sa    }
 { ajung la acest nivel prin oricare din  }
 { blocurile existente la nivelul inferior }
    for i:=0 to w[h]-1 do
      for j:=0 to w[h]-1 do
        AjungeIn(h-1,i+1,j+1,h,i,j,
   min[Offset(h-1,i+1,j+1)]+d[Offset(h,i,j)])
  end
end;

procedure AfiseazaDrum;
            { afiseaza drumul minim gasit }
var dh,dx,dy:piramida_byte;{Stocheaza drumul}
    h,x,y,pos,k:Integer;
begin
  k:=0;    { Numar de blocuri prin care a  }
           { trecut drumul                 }
  h:=m-1;  { Pozitia blocului curent h,x,y }
  x:=0; y:=0;
  while h<100 do
  begin
    dh[k]:=h;       { Stochez acest bloc si }
          { incrementez numarul de blocuri }
    dx[k]:=x; {prin care trece drumul minim}
    dy[k]:=y;
    Inc(k);
    pos:=Offset(h,x,y);
    h:=hback[pos]; {Trec la blocul anterior}
    x:=lback[pos]; y:=cback[pos]
  end;
  Writeln(fout,' ',k);
              { Afisez lungimea drumului }
  for h:=k-1 downto 0 do
           { Afisez in sens invers drumul }
  Writeln(fout,dh[h],' ',dx[h]+1,' ',dy[h]+1)
end;
```

```
Begin
  Assign(fin,'PIRAMIDA.IN');
  Reset(fin);
                          { Fisier de intrare }
  Assign(fout,'PIRAMIDA.OUT');
  Rewrite(fout);
                          { Fisier de iesire }
  Readln(fin,n);  { Citirea lungimii bazei }
  Initializare;
     { Procedura de initializare variabile }
  for k:=0 to m-1 do
    for i:=0 to w[k]-1 do
    begin
      for j:=0 to w[k]-1 do
      Read(fin,d[Offset(k,i,j)]);
                    { Citirea duritatilor }
      Readln(fin)
    end;
  for i:=0 to m-1 do
    CalculNivel(i);
              { Calcularea drumurilor pana }
              { la un fiecare nivel        }
  Write(fout,min[Offset(m-1,0,0)]);
         { Scrierea lungimii minime gasite }
  AfiseazaDrum;
                { Afisarea drumului gasit }
  Close(fin);
  Close(fout)
End.
```

P079807: Rigle Golomb

Problemă propusă de conf. dr. Henri Luchian, Universitatea "Al. I. Cuza", Iași

În loc de analiza riguroasă a problemei vă vom "istorisi" cele întîmplate la Constanța. "Ideile" olimpicilor pot fi utile ori de câte ori veți afla în fața unei probleme având aceleași caracteristici.

Aceasta a fost o problemă la care aproape toți concurenții au reușit să "păcălească" juriul. Problema a fost considerată ca fiind destul de dificilă, comisia așteptându-se la soluții "speciale" din partea concurenților cum ar fi cele bazate pe algoritmii genetici. Nu a fost să fie așa, deoarece comisia a scăpat din vedere faptul că datele de intrare conțineau un singur număr și acesta putea lua valori între 1 și 16. Din această cauză nu existau decât 16 posibilități pentru rezultate, iar fișierul de ieșire nu conținea decât maxim 16 numere. Așa că, spre mai mica sau mai marea surprindere a comisiei, concurenții au declarat niște constante care erau afișate în funcție de valoarea care se găsea in fișierul de intrare. Singurul impediment în obținerea punctajului maxim era determinarea celor 16 soluții posibile. Concurenții au folosit diferite metode pentru a le determina în cele 4 ore pe care le-au avut la dispoziție. S-au folosit algoritmi probabiliști (dar nu genetici), metode backtracking, metode euristice etc. Au rulat programele respective în "background" și au copiat rezultatele în constantele declarate în programele predate pentru evaluare. (Se pare că multi-taskingul din Windows poate fi folosit la ceva!) Și pentru ca participanții să nu se plângă că problema a fost prea dificilă, cea mai mare valoare cu care au fost testate programele a fost 11 (unsprezece). Mai mult, s-a acordat punctaj maxim chiar dacă soluția nu era optimă, ci doar relativ apropiată de ceea ce ar fi trebuit să se obțină, așa că unii concurenți au obținut cele 50 de puncte cu problema rezolvată doar pe jumătate. Mai trebuie precizat faptul că olimpicii nu au ezitat să-și arate "recunoștința" față de generozitatea comisiei și și-au manifestat dorința de a mai rezolva câteva probleme de acest fel. Având în vedere cele precizate mai sus, programul următor ar fi obținut cu siguranță punctajul maxim.

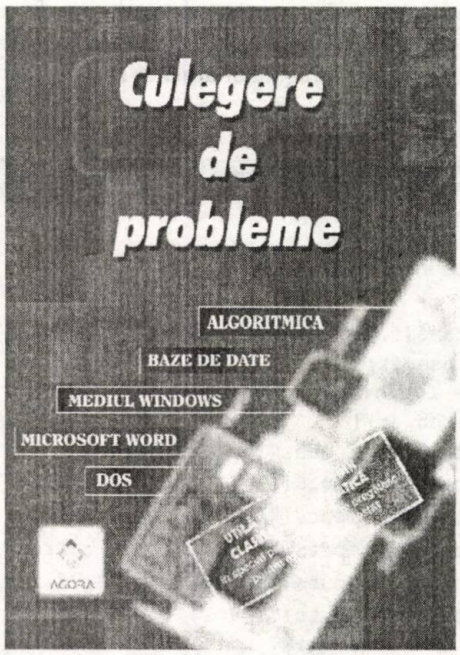
Listing RIGLA.CPP

```
#include <stdio.h>
void main(void){
  FILE *f=fopen("GOL.IN","rt");
  int n;
  fscanf(f,"%d",&n);
  fclose(f);
  f = fopen("gol.out", "wt");
  switch (n){
    case 1: fputs("1", f); break;
    case 2: fputs("1 3", f); break;
    case 3: fputs("1 4 6", f); break;
    case 4: fputs("1 4 9 11", f); break;
    case 5: fputs("1 4 10 15 17", f); break;
    case 6: fputs("1 4 10 18 23 25", f);
            break;
    case 7: fputs("1 4 9 15 22 32 34", f);
            break;
    case 8: fputs("1 5 12 25 27 35 41 44", f);
            break;
    case 9: fputs("1 6 10 23 26 34 41 53 \
55", f); break;
    case 10: fputs("1 4 13 28 33 47 54 64 \
70 72", f); break;
    case 11: fputs("2 6 24 29 40 43 55 68 \
75 76 85", f); break;
    case 12: fputs("2 5 25 37 43 59 70 85 \
89 98 99 106", f); break;
    case 13: fputs("1 6 14 21 23 39 49 73 \
76 105 117 136 147", f); break;
    case 14: fputs("1 4 13 20 28 30 51 65 \
85 108 119 141 175 181", f); break;
    case 15: fputs("1 3 9 19 33 50 55 78 93 \
118 122 157 184 205 218", f); break;
    case 16: fputs("1 3 11 23 36 50 55 79 \
96 117 124 154 188 194 236 251", f); break;
}
fclose(f);
}
```

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie* ... un MANUAL.



Culegerea conține:

- probleme de algoritmică, majoritatea rezolvate în Pascal
- probleme de gestiune a bazelor de date (FOX-PRO)
- probleme care verifică îndemânarea de a lucra în mediul Windows și cu editorul Microsoft Word
- probleme de verificare a cunoștințelor aferente sistemului de operare DOS

Această culegere constituie un sprijin pentru colectivele de profesori și pentru elevii care doresc să pregătească probleme pentru atestatul de informatică dar poate fi folosită și în scopul pregătirii pe parcursul celor patru ani de liceu.

Culegerea a fost realizată de un număr mare de profesori cu o bogată experiență în predarea informaticii la toate nivelurile. Astfel, culegerea conține o varietate mare de enunțuri de probleme rezolvate, abordând diferite metode și stiluri de programare.

• • •

Dacă vreți, considerați că participați la o întrecere și încercați să rezolvați aceste probleme (suntem convinși că puteți) mai elegant și cu o eficiență în programare mai ridicată.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422] sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

cu precizarea că era mai natural ca invocarea metodei statice `met` să se fi făcut prin `Clasa.met();` adică fără a mai crea o instanță `Ob` a clasei.

Menționăm însă că în general modificatorul **`final`** este folosit, însoțit de **`static`**, pentru a trata un câmp ca o "constantă cu nume".

Efectul modificatorului **`final`** asupra metodelor și claselor va fi prezentat atunci când vom vorbi despre extinderea claselor.

Modificatorul static

Modificatorul **`static`** poate fi folosit la declararea câmpurilor și metodelor unei clase; cuvântul cheie **`static`** mai este folosit și pentru a anunța un bloc de inițializare **`static`**.

O variabilă locală nu poate fi declarată cu **`static`**.

Fie `c` un câmp declarat cu **`static`** în clasa `Clasa`. Această declarare are două consecințe:

- câmpul `c` este comun tuturor obiectelor de tipul `Clasa`;
- referirea la câmp din exteriorul clasei se face fie conform regulii generale (prin crearea și utilizarea unei instanțe a clasei `Clasa`) fie direct, prin precalificarea câmpului cu numele clasei (`Clasa.c`).

Câmpurile statice sunt inițializate înainte de a fi folosite și înainte ca vreo metodă a clasei respective să fie executată.

Fie acum `met` o metodă declarată cu modificatorul **`static`** în clasa `Clasa`; ea se numește *metodă statică* sau *metodă de clasă*. Invocarea ei se poate face fie conform regulii generale (prin crearea și utilizarea unei instanțe a clasei `Clasa`) fie direct, prin precalificarea metodei cu numele clasei: `Clasa.met(...);` de aceea în metodă nu poate fi folosită referința **`this`**.

O metodă statică poate accesa numai câmpurile și metodele statice ale clasei. Metodele statice (de clasă) sunt folosite de obicei pentru a efectua prelucrări asupra câmpurilor statice ale clasei. Putem "ocoli" însă această regulă, transmițând (ca parametru) metodei o referință la un obiect, dar acest mod de lucru apare ca nenatural.

Lucrul cu blocuri de inițializare statice a fost descris într-un paragraf precedent.

Modificatorii de acces

Toate câmpurile și metodele unei clase sunt accesibile (pot fi referite) din interiorul clasei.

Rolul modificatorilor **`public`**, **`protected`**, **`private`** și cel implicit (numit uneori *package*) referitor la accesul din exteriorul unei clase la membrii săi este:

Modificator	*Membrii clasei sunt accesibili*
`public`	- de oriunde clasa este accesibilă
implicit (*package*)	- din cod din același pachet
`protected`	- din cod din același pachet
`private`	- numai din interiorul clasei

```
public static void main(String[] s) {
  System.out.println( nrapar(s[0],s[1]));
 }
}
```

Compararea șirurilor

Clasa `String` prevede metode pentru compararea conținutului șirului care invocă metoda conținutul unui șir transmis ca parametru. Vom presupune că șirul care invocă metodele este `s`. Compararea se referă la șirurile în totalitatea lor sau la subsecvențe ale lor.
Compararea se face conform ordinii lexicografice; compararea a două caractere constă în compararea numerelor lor de ordine în setul de caractere Unicode.

`boolean` `equals(String s1)`
: întoarce `true` dacă șirurile coincid;

`int CompareTo(String s1)`
: întoarce o valoare negativă (dacă `s<s1`), zero (dacă `s=s1`) sau o valoare pozitivă (dacă `s>s1`);

`boolean` `regionMatches(int p,String s1, int p2,int lung)`
: compară secvențele de lungime `lung` ce încep în `s` de pe poziția `p`, respectiv în `s1` de pe poziția `p1`; este întoarsă valoarea `true` dacă și numai dacă aceste secvențe coincid;

`boolean` `startsWith(String pre)`
: întoarce `true` dacă șirul `pre` este prefix al lui `s` și `false` altfel;

`boolean` `endsWith(String suf)`
: întoarce `true` dacă șirul `suf` este sufix al lui `s` și `false` altfel.

Să remarcăm că metoda `equals` nu poate fi suplinită de operatorul `==` deoarece prin `s==s1` se compară două referințe și nu conținutul lor!

Determinarea unui subșir cu o anumită proprietate

Metodele clasei `String` ce se încadrează aici sunt următoarele (vom presupune din nou că șirul care invocă metoda este notat prin `s`):

`String substring(int p)`
: este întors subșirul lui `s` ce începe de pe poziția `p`;

`String substring(int p, int u)`
: este întors subșirul lui `s` ce începe de pe poziția `p` și se termină pe poziția `u-1`;

`String replace(char old, char new)`
: este întors șirul obținut din `s` prin înlocuirea tuturor caracterelor `old` prin caracterul `new`;

`String toUpperCase()`
: fiecare caracter este înlocuit cu echivalentul său majusculă, dacă acesta există;

`String toLowerCase()`
: fiecare "literă mică" este înlocuită cu "literă mare",

dacă această corespondență există;

`String trim()`

întoarce șirul obținut din `s` prin eliminarea spațiilor albe de la începutul și sfârșitul său;

`String concat(String s1)`

întoarce șirul `s+s1`.

Șiruri și tablouri de caractere

Clasa `String` prevede metode pentru transformarea șirurilor în tablouri de caractere, precum și pentru transformarea inversă:

Metoda statică `toCharArray()` invocată de un șir `s` întoarce un tablou de caractere de aceeași lungime și cu aceleași elemente.

Metoda statică `copyValueOf` are două forme:

```
String copyValueOf(char[] a)
String copyValueOf(char[] a, int p,
                                int lung)
```

și întoarce șirul corespunzător tabloului `a` de caractere, respectiv numai acea secvență din tablou ce începe pe poziția `p` și are lungimea `lung`.

De exemplu, pentru a determina șirul obținut din șirul `s` prin considerarea elementelor sale în ordine inversă, putem scrie următoarea metodă:

```
public static String invers(String s) {
  char c;
  char[] a=s.toCharArray();
  for (int p=0,u=a.length-1; p<u; p++,u--)
  {
    c=a[p]; a[p]=a[u]; a[u]=c;
  }
  return String.copyValueOf(a);
}
```

O altă metodă pentru compararea șirurilor

Metoda `String intern()` întoarce un șir având același conținut ca cel al șirului care invocă metoda, având în plus următoarea proprietate: *pentru orice două șiruri cu același conținut, valoarea întoarsă de metoda* `intern` *este aceeași*. Drept urmare putem verifica dacă două șiruri au același conținut prin:

`s1.intern()==s2.intern()` *care este mai rapidă decât:* `s1.equals(s2)`.

Exemplu

De la intrare sunt citite cel mult 100 de șiruri de caractere delimitate de ghilimele. Vrem să reținem doar pe cele distincte, urmând ca în final să le listăm.

```
import java.io.*;
class sirunic {
  static String vect[] = new String[100];
```

Reamintim că modificatorii de mai sus joacă un rol nu numai în privința accesului, dar și pentru facilitatea de extindere a claselor, ceea ce diferențiază modificatorul **protected** și cel implicit. Evident, vom reveni asupra acestor modificatori când vom studia extinderea claselor.

Determinarea semnificației unui identificator

Pentru un identificator folosit ca nume de variabilă (locală sau câmp), metodă sau clasă, semnificația sa se obține la prima căutare încununată de succes din următorul șir de căutări succesive:

1) variabilele locale declarate într-un bloc sau un ciclu **for**;
2) numai dacă suntem în interiorul unei metode sau a unui constructor: parametrii metodei sau constructorului;
3) elemente rezultate prin extinderea claselor (vom face precizări la momentul respectiv);
4) tipuri importate explicit;
5) tipuri declarate în același pachet;
6) tipuri importate implicit;
7) pachete disponibile pe sistemul gazdă.

Mai mult, se face distincție clară între tipurile de identificatori: nume de variabile (locale), de pachete, de clase, de metode și de câmpuri; iată de ce numele unei metode nu poate fi ascuns. Aceasta permite să folosim un același identificator pentru a numi una sau mai multe astfel de entități. De exemplu, următoarea clasă este corectă din punct de vedere sintactic, compilatorul nesemnalând vreo eroare:

```
class obsesie {
  obsesie obsesie; obsesie()
  { obsesie = new obsesie(); }
  obsesie obsesie (obsesie obsesie) {
    obsesie:
    while (obsesie != null)
      if (obsesie.obsesie(obsesie)==obsesie)
        break obsesie;
  return obsesie;
  }
}
```

Nu recomandăm să se abuzeze astfel de distincția menționată. Clasa de mai sus poate fi scrisă sub următoarea formă mai clară și "neobsesivă":

```
class obsesie {
  obsesie C;
  obsesie() { C = new obsesie(); }
  obsesie met(obsesie param){
    et:
    while (param!=null) {
      if (param.met(param)==param) break et;
    }
    return param;
  }
}
```

```
  static int lung;
  static void adaug(String s) {
    String martor = s.intern();
    for (int i=0; i<lung; i++)
      if (vect[i]==martor) return;
    vect[lung++] = martor; return;
  }

  public static void main (String[] str)
                          throws IOException {
    StreamTokenizer f=new
    StreamTokenizer(System.in);
    f.nextToken();
    while (f.ttype=='\"') {
      adaug(f.sval); f.nextToken();
    }
    for (int i=0; i<lung; i++)
      System.out.println(vect[i]);
  }
}
```

Este important de notat că programul funcționează corect deoarece se lucrează doar cu șiruri obținute ca efect al invocării metodei `intern`.

Conversii

În multe cazuri suntem puși în situația de a converti valori numerice în șiruri sau de a lucra cu o valoare numerică memorată într-un șir.

Șirul ce corespunde unei valori numerice sau booleene `x` se obține simplu prin: `String.valueOf(x)`

Fie acum `s` un șir ce conține un literal boolean sau numeric `x`. Atunci x se obține prin:

`new Boolean(s).booleanValue()`, pentru tipul `boolean`;
`new Float(s).floatValue()`, pentru tipul `float`;
`new Double(s).doubleValue()`, pentru tipul `double`;
`Integer.parseInt(s,b)`, pentru tipul `int`;
`Long.parseLong(s,b)`, pentru tipul `long`.
cu mențiunea că atunci când conținutul șirului nu are forma așteptată, este lansată excepția `NumberFormatException`.

Cele de mai sus nu se aplică pentru valori scrise în octal sau hexazecimal (cu prefixele `0`, respectiv `0x`).

Metoda `valueOf` a clasei `String` mai poate fi invocată și pentru un caracter; va fi întors șirul format din acel unic caracter.

Prof. dr. Horia Georgescu este cadru didactic al Facultății de Matematică a Universității din București. E-mail: hg@colegiu.cs.unibuc.ro

Premiul lunii

Începând din această lună, revista noastră va încerca să ofere în fiecare număr câte un *PREMIU DE FIDELITATE* unui cititor ales prin tragere la sorți dintre *ABONAȚI*.

Și tu ai șansa să câștigi *PREMIUL LUNII*. Condiția este simplă: *ABONEAZĂ-TE !*

Premiul lunii noiembrie este oferit de sponsorul nostru, *MICROSOFT ROMANIA*...

... și a fost câștigat de către

RADU CUCU din Caransebeș.

Pagini WEB cu HTML 3.2 (II)

Mihai Scorțaru

Cele mai importante și interesante elemente ale unei pagini Web sunt imaginile, 99,9% din paginile Web existente conținând imagini. Dar timpul trece... și imaginea nu se încarcă... Cum să evităm ca vizitatorul paginii noastre să nu apese butonul STOP, pierzându-și răbdarea?

Amplasarea unei imagini într-o pagină Web

Deși au o importanță mare, imaginile conținute în paginile Web constituie și o mare problemă din cauza timpului relativ mare necesar încărcării lor în timpul navigării pe Internet. Majoritatea browserelor conțin o opțiune de tipul *Auto Load Images*. Prin deselectarea acestei opțiuni browserul va fi informat că nu se dorește încărcarea automată a imaginilor în același timp cu textul. În locul unei imagini va apărea un icon, însoțit eventual de un text. Dacă se dorește încărcarea unei imagini trebuie selectat icon-ul corespunzător. Opțiunea de tipul *Auto Load Images* diferă de la un browser la altul și de la o versiune la alta. De exemplu, în Internet Explorer 4.0, opțiunea corespunzătoare este:

View | Internet Options... | Advanced | Multimedia | Show Images.

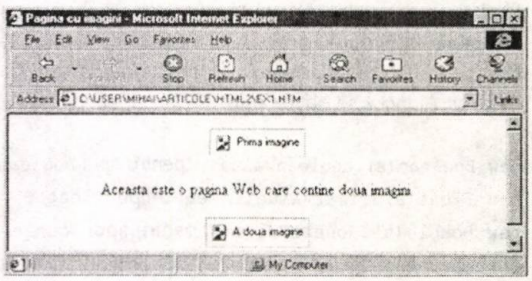
În Internet Explorer 4.0. icon-ul care apare de obicei este:

Tag-ul HTML cu ajutorul căruia se afișează o imagine este **<IMG>**. Cei mai importanți parametri ai săi sunt: **SRC** și **ALT**. Cu ajutorul parametrului SRC se precizează fișierul în care este conținută imaginea, iar cu ajutorul parametrului ALT se precizează un text alternativ care va apărea în cazul în care imaginea nu poate fi încărcată din diferite motive cum ar fi: formatul nu este suportat de browser, fișierul respectiv lipsește sau este corupt, este deselectată opțiunea de tipul *Auto Load Images* etc.

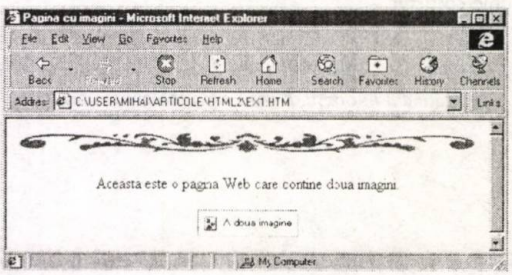
Să considerăm fișierul HTML:

```
<html>
<head>
<title>Pagina cu imagini</title>
</head>
<body>
<div align = "center">
<img src="img1.gif" alt="Prima imagine">
<br><br>Aceasta este o pagina Web care contine
doua imagini.<br><br>
<img src="img2.gif" alt="A doua imagine">
</div>
</body>
</html>
```

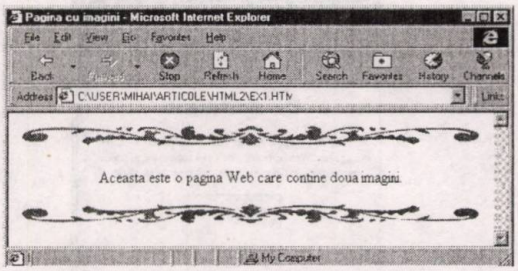
În cazul în care imaginile nu pot fi încărcate sau nu este dorită încărcarea lor, pagina Web corespunzătoare va arăta astfel:



Pentru a cere afișarea unei imagini, comenzile diferă de la un browser la altul. Pentru *Netscape Navigator* nu trebuie decât să efectuați un left-click pe icon-ul care a fost afișat în locul imaginii. Pentru *Internet Explorer 4.0* trebuie efectuat un right-click pe icon-ul corespunzător și din meniul pop-up care va apărea trebuie selectată comanda *Show picture*. Dacă vom cere încărcarea primei imagini și această cerință va putea fi îndeplinită, va fi afișată următoarea pagină:



Se observă că a doua imagine nu este încărcată automat atunci când se cere încărcarea primei imagini. Pentru a vedea și cealaltă imagine trebuie să efectuăm aceleași operații ca și pentru prima. În cazul în care a doua imagine este disponibilă ea va fi încărcată, iar pagina Web corespunzătoare va arăta astfel:



Trebuie precizat că, în cazul în care cele două imagini ar fi fost identice (atributul `SRC` al tag-ului `IMG` ar fi conținut același fișier) atunci, dacă s-ar fi cerut încărcarea uneia din cele două imagini, ar fi fost afișate amândouă.

Alinierea imaginilor

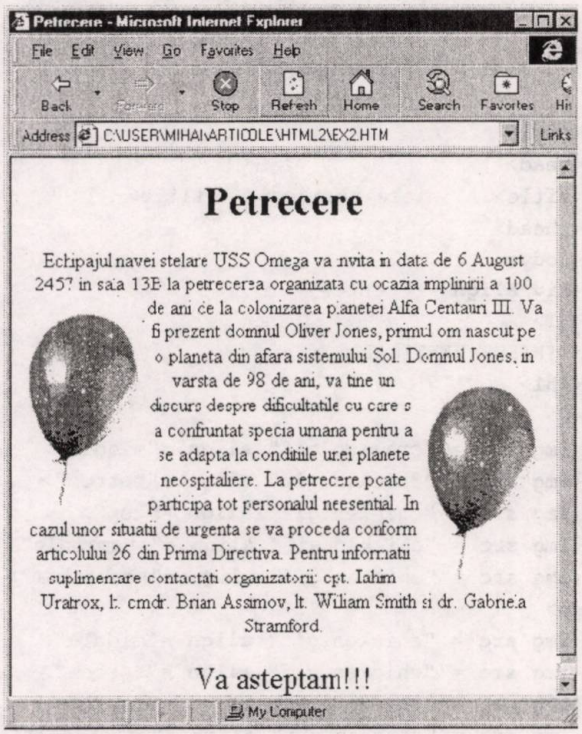
Tag-ul html `<IMG>` mai are un parametru important și anume `ALIGN` cu ajutorul căruia pot fi aliniate imaginile. O imagine poate fi aliniată atât orizontal, cât și vertical.

Pentru alinierea orizontală, valorile parametrului **`ALIGN`** pot fi `left`, `right` sau `center`. Pentru alinierea verticală valorile pot fi: `top`, `bottom`, `absbottom`, `middle` sau `absmiddle`.

În cazul alinierii orizontale imaginea va fi aliniată la dreapta, la stânga sau va fi centrată. Trebuie observat faptul că parametrul **`ALIGN`** al comenzii **`IMG`** diferă de același parametru al comenzii **`DIV`**. De exemplu `<IMG ALIGN = "right">` va alinia o imagine la dreapta și textul care urmează va "înfășura" imaginea. Acest fapt este ilustrat în următorul exemplu:

```
<html>
<head>
<title> Petrecere </title>
</head>
<body>
<div align="center">
<h1> Petrecere </h1>
   Echipajul navei stelare USS Omega va
invita in data de 6 August 2457 in sala 13B
la petrecerea organizata cu
<img src = "balon.gif" alt = "balon" align =
"left">
ocazia implinirii a 100 de ani de la
colonizarea planetei Alfa Centauri III. Va
fi prezent domnul Oliver Jones, primul om
nascut pe o planeta din afara sistemului
Sol.
<img src = "balon.gif" alt = "balon" align =
"right">
Domnul Jones, in varsta de 98 de ani, va
tine un discurs despre dificultatile cu care
s-a confruntat specia umana pentru a se
adapta la conditiile unei planete
neospitaliere. La petrecere poate participa
tot personalul neesential. In cazul unor
situatii de urgenta se va proceda conform
articolului 26 din Prima Directiva. Pentru
informatii suplimentare contactati
organizatorii: cpt. Iahim Uratrox, lt. cmdr.
Brian Assimov, lt. William Smith si dr.
Gabriela  Stramford.
<p>
<basefont size=5>
Va asteptam!!!
</div>
</body>
</html>
```

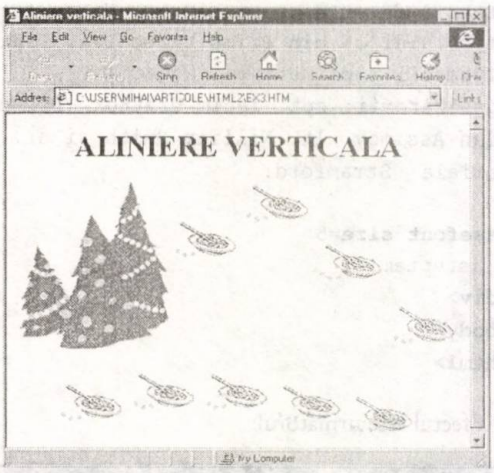
Efectul este următorul:



Valorile posibile ale parametrului **`ALIGN`** pentru alinierea verticală sunt: `top`, `bottom`, `absbottom`, `middle` și `absmiddle`. Dacă se folosește valoarea `top` marginea superioară a imaginii va fi aliniată cu marginea superioară a celei mai "înalte" imagini sau litere de pe linia pe care apare imaginea. Pentru a alinia marginea inferioară a imaginii cu marginea inferioară a textului folosiți valoarea `bottom`. Dacă doriți ca marginea inferioară a imaginii să fie aliniată

cu marginea inferioară a celei mai "de jos" imagini, trebuie folosită valoarea absbottom. Valoarea middle este folosită pentru a alinia mijlocul imaginii cu mijlocul textului. Pentru a alinia mijlocul imaginii cu centrul absolut a tot ceea ce apare pe linie, folosiți valoarea absmiddle.

Pe pagina următoare veți vedea o pagină Web care ilustrează folosirea alinierii verticale:



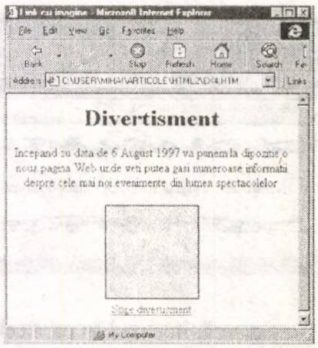
Fișierul HTML corespunzător este:

```
<html>
<head>
<title> Aliniere verticala </title>
</head>
<body>
<div align = "center">
<h1>
ALINIERE VERTICALA
</h1>
<P>
<img src = "craciun.gif" align ="middle">
<img src = "chicken.gif" align ="bottom">
<img src = "chicken.gif" align ="top">
<img src = "chicken.gif" align ="absmiddle">
<img src = "chicken.gif" align ="absbottom">
<p>
<img src = "chicken.gif" align ="middle">
<img src = "chicken.gif" align ="bottom">
<img src = "chicken.gif" align ="top">
<img src = "chicken.gif" align ="absmiddle">
<img src = "chicken.gif" align ="absbottom">
</div>
</body>
</html>
```

Se observă că, datorită faptului că prima imagine din primul rând este foarte mare, imaginile mici din primul rând au "diferențe de nivel" mult mai mari decât cele din al doilea rând.

Imagini care sunt link-uri

Dacă se dorește ca un link să conțină și o imagine grafică acest lucru poate fi realizat prin includerea unei comenzi <IMG> între opening tag-ul și closing tag-ul comenzii <A>. Un astfel de link poate fi văzut în imaginea următoare:



Indiferent dacă veți apăsa pe baloane sau pe textul de sub ele browser-ul va căuta fișierul **DIVERTISMENT.HTML** și va încerca să afișeze pagina Web corespunzătoare. Fișierul HTML care are acest efect este următorul:

```
<html>
<head>
<title> Link cu imagine </title>
</head>
<body>
<div align = "center">
<h1> Divertisment </h1>
Incepand cu data de 6 August 1997 va punem
la dipozitie o noua  pagina Web unde veti
putea gasi numeroase informatii despre cele
mai noi evenimente din lumea spectacolelor.

<p>
<a href = "divertisment.html">
<img src = "baloane.jpg"> <br>
Spre divertisment
</a>
</body>
</html>
```

Formate de imagini suportate de browserele HTML

Dacă un browser respectă standardul HTML atunci el va recunoaște cu siguranță formatele GIF și JPEG. Unele browsere pot recunoaște și alte formate cum ar fi BitMap, PCX etc., dar nu se recomandă folosirea unor imagini în aceste formate deoarece majoritatea browserelor nu le vor putea afișa.

Micșorarea timpului necesar pentru încărcarea unei pagini

În majoritatea cazurilor, alături de fișierul .HTM sau .HTML sunt încărcate multe alte fișiere care conțin diferite infor-

mații. Cele mai frecvente astfel de fișiere sunt cele care conțin imagini. Datorită numărului mare de octeți care trebuie încărcați, *intervalul de timp dintre momentul în care este comandată afișarea unei imagini și momentul în care ea este afișată complet, este destul de mare*. Timpul necesar încărcării unei pagini este foarte important, deoarece un navigator pe Internet nu va aștepta prea mult ca o pagină să se încarce și după un interval de timp mai mic sau mai mare va renunța la vizitarea acestei pagini. Bineînțeles, nici un proiectant de pagini Web nu dorește ca butonul STOP al browserului să fie apăsat în timp ce pagina sa este încărcată.

O practică des întâlnită este micșorarea numărului de culori care apar în imagine la 256 sau 16. Aceasta va reduce considerabil dimensiunea fișierului care conține imaginea și acel fișier va putea fi încărcat mult mai repede.

O altă modalitate este folosirea unor formate GIF sau JPEG speciale numite Interlaced GIF respectiv Progressive JPEG. Dacă o imagine este în acest format atunci la început se va afișa o imagine de calitate inferioară și după ce restul paginii se încarcă, este afișată imaginea întreagă. Folosirea acestor formate are un profund efect psihologic, deoarece navigatorul poate să vadă o imagine completă care se va îmbunătăți treptat și nu trebuie să aștepte pentru o imagine de calitate superioară care ar apărea deodată pe ecran.

Mărimea unei imagini

Deoarece textul este încărcat mult mai repede decât imaginile, cele mai multe browsere Web vor afișa textul înaintea imaginilor. La început browserul nu cunoaște dimensiunile imaginilor și de aceea nu va lăsa loc liber pentru ele. Din această cauză textul se va mișca pe ecran în timp ce imaginile sunt încărcate. Pentru a evita acest lucru comanda <IMG> are doi parametri prin care poate fi precizată *lungimea* și *înălțimea* unei imagini. De exemplu, comanda:

```
<IMG SRC="myimage.jpg" WIDTH=200 HEIGHT=100>
```

va rezerva un dreptunghi de dimensiuni 200x100 pentru imaginea din fișierul **myimage.gif**. Dimensiunile transmise ca parametri nu trebuie neapărat să fie egale cu dimensiunile reale ale imaginii, browser-ul mărind sau micșorând imaginea astfel încât ea să ocupe întreg dreptunghiul precizat. Folosirea acestor parametri poate avea rezultate neplăcute deoarece imaginea ar putea fi deformată prin mărirea sau micșorarea disproporționată a celor două dimensiuni.

Imagini Preview

Un alt parametru al comenzii <IMG> este **LOWSRC**. Acesta oferă posibilitatea ca la început să fie afișată o imagine, iar după încărcarea restului paginii această imagine să fie înlocuită cu alta de calitate superioară. Un exemplu ar fi:

```
<IMG SRC="highqlty.jpg" LOWSRC="lowqlty.jpg">.
```

De obicei imaginea "preview" este un JPEG cu un grad de compresie foarte mare sau o imagine care conține un număr foarte mic de culori (16, 4 sau 2).

Animații grafice

Metodele avansate de introducere a animațiilor în paginile Web vor fi tratate în unul din episoadele următoare. Există totuși o metodă simplă de a include animații într-o pagină Web și anume folosirea GIF-urilor animate care pot fi introduse la fel de ușor ca orice altă imagine.

Culori

În mod normal o pagină Web are niște culori predefinite pentru fond, text și link-uri. Aceste culori pot fi schimbate cu ajutorul unor parametri ai tag-ului **BODY**. O comandă:

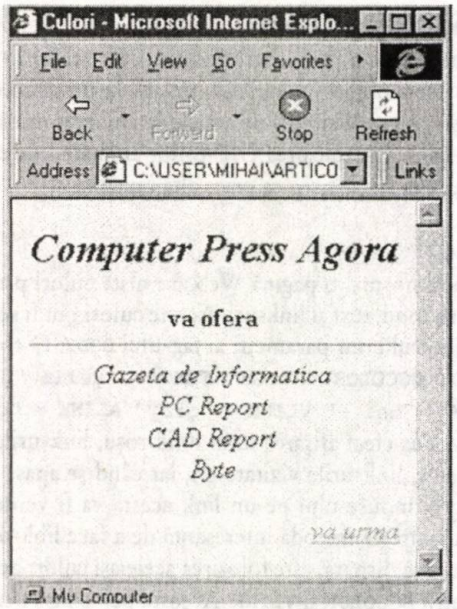
```
<BODY BGCOLOR = "red" TEXT = "yellow"
LINK = "white" VLINK = "gray" ALINK = "green">
```

va avea ca efect afișarea unui fond roșu, link-urile nevizitate albe, link-urile vizitate gri, iar când se apasă butonul stâng al mouse-ului pe un link acesta va fi verde pentru scurt timp. O metodă interesantă de a face link-urile deja vizitate să dispară, este folosirea aceleiași culori pentru parametrii **BGCOLOR** și **VLINK**. Aceasta poate însă nemulțumi unii navigatori care ar dori să revadă o pagină deja vizitată.

Dacă valoarea unui parametru este de tipul #RRGGBB, atunci va fi definită o nouă culoare. RR, GG și BB sunt numere hexazecimale de două cifre și reprezintă valorile pentru componenta roșie, verde și albastră a culorii. De exemplu, un roșu deschis va corespunde valorii #FF0000, verde închis va fi #004000, iar un gri de intensitate medie va fi #808080. Iată un exemplu care folosește noțiunile prezentate anterior:

```
<html>
<head>
<title> Culori </title>
</head>

<body bgcolor = "#FFFF80" text = "#400040"
link = "#FF8000" vlink ="#FFFF80">
<div align = "center">
<h2> <i>
Computer Press Agora
</i> </h2>
<b> va ofera </b>
<br> <hr>
<i> Gazeta de Informatica <br>
PC Report <br>
CAD Report <br>
Byte <br>
</div>
<p align = "right">
<a href = "next.htm"> va urma </a>
</body>
</html>
```

La încărcarea acestui fișier va apărea următoarea pagină Web:

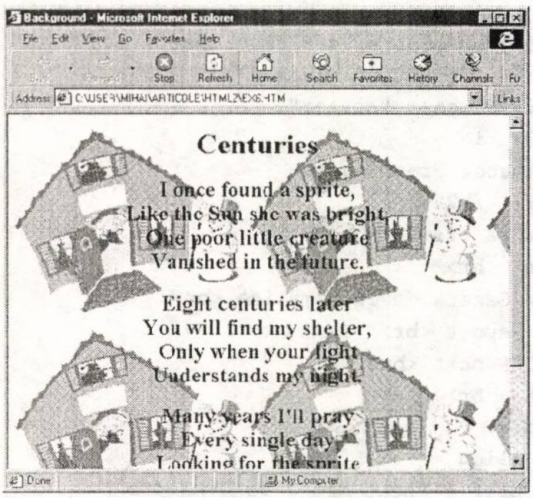


Imagini Background

Majoritatea paginilor Web au ca fond o anumită imagine. Specificarea unei imagini ca *background* se poate face cu ajutorul parametrului **BACKGROUND** al *tag*-ului **<BODY>**. Sintaxa este:

```
<BODY BACKGROUND = "background.jpg">.
```

Dacă imaginea nu va acoperi întreaga fereastră în care este afișată pagina Web, părțile libere vor fi acoperite cu copii ale imaginii. O pagină Web cu background este următoarea:



Fișierul HTML corespunzător este:

```
<html>
<head>
<title> Background
</title>
</head>
<body background = "back.gif">
<div align = "center">
<b> <basefont size = 5>
<h1> Centuries </h1>
I once found a sprite,<br>
Like the Sun she was bright,<br>
One poor little creature<br>
Vanished in the future.<p>
Eight centuries later<br>
You will find my shelter,<br>
Only when your light<br>
Understands my night.<p>
Many years I'll pray<br>
Every single day<br>
Looking for the sprite<br>
I have lost that night.<p>
Nine centuries later<br>
Delivering a letter<br>
Another sprite I found.<br>
</b>
</div>
</BODY>
</HTML>
```

Dacă veți folosi imagini background, trebuie să aveți grijă ca textul care va fi afișat peste background să fie vizibil. Încercați să folosiți pentru text o culoare care nu apare deloc în background sau care apare de foarte puține ori.

În loc de concluzie

După parcurgerea acestor două episoade ale serialului puteți proiecta deja o pagină Web care să-și merite numele. În episoadele următoare vor fi prezentate câteva modalități care vor îmbunătăți calitatea paginilor dumneavoastră. Veți afla noutăți despre introducere de secvențe video, tabele, *frame*-uri, sunete, *script*-uri etc. Sperăm că vă vor interesa *applet*-urile care pot da o notă de profesionalism unei pagini Web.

Mihai Scorțaru este student în anul I al Facultății de Automatică și Calculatoare, Universitatea Tehnică Cluj. Poate fi contactat prin e-mail la adresa: scort@licj.soroscj.ro

```
Set Doc=ActiveDocument
Set myRange=Doc.Range(100,120)
myRange.InsertAfter "Vulpea care nu ajunge la
 +"struguri zice că-s acrii, pentru ca simte"
                          +"gustul insuccesului"
```

Imaginea lui Einstein

Un alt obiect Word este `Assistant` care referă asistentul folosit. Următoarea aplicație face ca asistentul curent să fie imaginea cu Einstein:

```
Sub asistent()
  Application.Assistant.FileName="Genius.act"
  Application.Assistant.Visible=true
End Sub
```

Macrouri dăunătoare

Virușii de documente sunt niște macrouri cu efecte dăunătoare pentru documente. Word-ul are câteva sute de macrouri gata definite. Aceste macrouri pot fi modificate și virusul este gata. De exemplu vom modifica macroul de `Save As`, astfel încât în momentul în care apăsăm acest buton să nu se mai realizeze salvarea documentului.

Pentru a vedea toate macrourile predefinite alegem `Tools | Macro | Macros`, iar din combobox-ul `Macros In` alegem itemul `Word Command`. Veți vedea o listă de vreo câteva sute de comenzi (dacă nu mă înșel sunt chiar 900) dintre cele mai variate. Pe noi ne interesează comanda `File Save As`. Selectăm această comandă, apoi din combobox-ul `Macros In` alegem itemul **`Normal.dot`** (Global Template) pentru a crea un macrou global. Apăsăm apoi pe butonul `Create`. Va apărea definiția acestui macrou:

```
Sub FileSaveAs()
' FileSaveAs Macro
' Saves the active document or template
  ActiveDocument.SaveAs
End Sub
```

Veți putea face orice fel de modificări. Dacă, de exemplu, puneți singura linie executabilă în comentariu, atunci degeaba veți apăsa butonul `SaveAs`, nimic nu se va întâmpla. Dacă doriți să refaceți vechea comandă `SaveAs`, ștergeți macroul pe care l-ați modificat.

Automatizarea

Am spus la începutul acestui articol că puteți accesa obiectele din Word nu numai folosind VBA, dar chiar folosind Delphi și C++Builder. În acest caz nu va mai fi vorba de un macrou, ci de o altă aplicație (Delphi sau C++Builder) care va manipula obiectele Word-ului. Antepenultimul cuvânt folosit în fraza anterioară este *a manipula*. V-aș ruga să fiți atenți la ceea ce am să spun în continuare despre acest cuvânt deoarece în jurul lui are să graviteze tot ceea ce urmează. Nu am să folosesc cuvântul *manipulare*, ci *automatizare* deoarece acesta este folosit și în documentația oficială a Microsoft-ului. Definiția acesteia (a *automatizării*) este *abilitatea unor aplicații de a manipula alte aplicații*. Mai precis, *automatizarea* este *abilitatea unor aplicații de a manipula obiecte aflate în alte aplicații*. Cei care automatizează se numesc *clienți*. Aplicațiile manipulate se numesc *servere*. Există o întreagă tehnologie referitoare la automatizare.

Piatra de hotar pe care este clădită tehnologia automatizării se numește COM (Component Object Model) care este de fapt un standard binar care permite diferitelor aplicații software să interacționeze între ele ca obiecte. Datorită faptului că standardul COM este binar, deci independent de un limbaj anume, aplicațiile scrise în diferite medii de programare precum VB, Delphi, C++Builder sau Visual C++ pot să-i folosească specificațiile și să interacționeze între ele. Din COM au derivat tehnologiile OLE și *ActiveX*, ultima dintre ele a fost folosită în campaniile de publicitate ale firmei Microsoft. De fapt în orice revistă de specialitate pe care o deschideți veți întâlni de câteva zeci, dacă nu chiar de câteva sute de ori acest X (care nu vine de la X-Files!).

Revenind la COM și OLE pot spune că totul se reduce la această artă a automatizării. Din moment ce o aplicație poate manipula o altă aplicație, întreg sistemul poate fi văzut ca nimic mai mult decât o suită de servicii pe care le puteți accesa când aveți nevoie. De exemplu, dacă aveți nevoie de un editor de text, accesați Word-ul, dacă aveți nevoie de operații cu baze de date manipulați Access-ul, dacă aveți nevoie de tabele manipulați Excel-ul. Toate aceste manipulări și încă altele se pot face din interiorul unei aplicații Delphi, C++Builder sau Visual C++.

Vă avertizez că aveți nevoie de un procesor destul de bunicel (Pentium) și cel puțin 16 Mb RAM. În caz contrar tehnologia vi se va părea cam suspectă, deoarece funcționează cam încet.

Manipularea Word-ului dintr-o aplicație Delphi

Există două posibilități de a face acest lucru: fie având Word-ul lansat în execuție, fie nu. În cel de al doilea caz chiar aplicația noastră va lansa în execuție Word-ul:

```
uses ComObj;
{$R *.DFM}
var v:Variant;
procedure TForm1.Button1Click(Sender: TObject);
begin
  V:=CreateOleObject('Word.Application');
          { Este lansat in executie Word-ul }
  V.Visible:=true; { Word-ul devine vizibil }
end;
```

În prima linie am lansat Word-ul în execuție, este vorba de o lansare invizibilă utilizatorului (veți auzi doar zgomotul produs de hard-discul propriu). Acesta (utilizatorul) trebuie să scrie și cea de a doua linie de program,

pentru ca Word-ul să devină vizibil. Nu insist asupra descrierii tipului `variant` și a funcției `CreateOleObject` deoarece aș depăși scopul propus în acest articol. Important este în acest moment să deprindeți niște mecanisme, pe care abia mai târziu le puteți înțelege. În principiu, în prima linie de program ați obținut o referință la aplicația Word (mai precis la un obiect aflat în interiorul acestei aplicații). Prin intermediul acestei referințe veți putea accesa metodele și proprietățile obiectului `Application`. De fapt acest lucru l-am făcut și când am scris linia `V.Visible:=true;`: am accesat proprietatea `Visible` a obiectului `Application`. Pentru o documentare completă asupra metodelor și proprietăților acestui obiect studiați fișierul help **`VBAWRD8.HLP`**.

Pentru a crea un document în Word veți scrie:

```
procedure TForm1.Button1Click(Sender:TObject);
begin
  V:=CreateOleObject('Word.Application');
  V.Visible:=true;
  V.Documents.Add;
end;
```

Aici am folosit o altă proprietate a obiectului Word, și anume `Documents` care reprezintă lista documentelor active la un moment dat într-o aplicație. Acest obiect are la rândul lui metoda `Add` care va adăuga un nou document vid la lista celor existente deja. Titlul lui va fi unul implicit. Pentru a deschide un fișier de pe disc scriem următorul cod:

```
procedure TForm1.Button1Click(Sender:TObject);
begin
  V:=CreateOleObject('Word.Application');
  V.Visible:=true;
  V.Documents.Open('c:\test.doc');
        { daca doriti sa salvati un fisier }
        { folositi metoda Save sau SaveAs  }
end;
```

Pentru a insera un text folosim obiectul `Range`:

```
procedure TForm1.Button1Click(Sender:TObject);
var v,r:Variant;
begin
  v:=CreateOleObject('Word.Application');
  v.Visible:=true;
  v.Documents.Add;
  r:=v.ActiveDocument.Range(0,0);
              { Inserarea se va realiza   }
              { la inceputul documentului }
  r.InsertAfter('Acesta este un test');
  v.ActiveDocument.SaveAs('c:\test.doc');
end;
```

Învățați... învățați...

Mă opresc aici cu prezentarea aplicațiilor care se pot realiza folosind posibilitatea manipulării Word-ului. Domeniul este extrem de vast și nu poate fi epuizat în câteva pagini. V-aș da un sfat de care sper să țineți cont, și anume: apucați-vă cât mai repede de programarea sub Windows și mai ales de facilitățile extraodinare pe care le oferă (printre care se numără și această posibilitate de manipulare a altor aplicații). Bineînțeles nu toate aplicațiile pot fi manipulate. Dar majoritatea celor create de către Microsoft au această facilitate pusă la dispoziție bineînțeles de către sistemul de operare.

În unul din numerele anterioare ale gazetei de informatică, la poșta redacției am remarcat afirmațiile unui cititor care spunea că programatorii de la Microsoft sunt "proști". Pentru a califica astfel pe cineva înseamnă ca tu însuți să fi mult mai deștept! Dar printr-o astfel de afirmație dovedești inferioritate... În plus, nu am auzit de nici un rezultat extraordinar al respectivului. Dacă este așa de deștept ar fi obținut cel puțin rezultatele lui *Radu Lupșa* (un om, având calități într-adevăr de excepție) care nu a făcut niciodată afirmații de genul celei amintite mai sus. (Radu Lupșa este absolvent al *Facultății de Matematică, secția Informatică al Universității Babeș-Bolyai* din Cluj; în prezent este beneficiarul unei burse pe care a câștigat-o prin concurs la *L'Ecole Normale Superieur* din Paris, în cadrul Facultății de Matematică - Informatică. La absolvire va obține o diplomă în *Studii Aprofundate în Algoritmică.*)

Considerându-mă în continuare în dialog cu respectiva persoană, aș mai aminti că am creat un DLL care compilat cu Delphi avea o lungime de patru ori mai mare decât compilat cu *Visual C++*.

Revenind la automatizarea aplicațiilor, v-aș sugera să o utilizați cât mai mult, deoarece aceasta este direcția (sau mai bine zis una din direcțiile) spre care se îndreaptă informatica actuală. Dacă nu o să o înțelegeți repede, atunci peste câțiva ani, când produsele software vor ajunge de ordinul Gigabyteților, vă va fi peste putință. Sincer să fiu, mă gândesc cu milă la informaticianul de mâine (și prin mâine înțeleg 4-5 ani) și la cantitatea de informație pe care trebuie să o parcurgă pentru ca să poată spune că deține cât de cât secretele unui domeniu destul de restrâns. Pe de altă parte ar fi de dorit să înțeleagă cât mai mulți că în viitor va trebui să-și formeze fiecare persoană un "capital" bogat de informații. Vor reuși cei care vor fi suficient de bine *informați.*

Am promis că vă voi prezenta situația în C++Builder la sfârșit dar nu cred că are rost să insist. Pentru cei care cunosc limbajul C++, e necesar să facă doar o translatare a instrucțiunilor din Pascal în C. Atenție însă, dacă folosiți C++Builder 1.0, fișierul pe care trebuie să îl includeți nu este **`ComObj.hpp`** ci **`OleAuto.hpp`** și **`Ole2.hpp`**.

Mihai Oltean este sudent la Facultatea de Matematică, Secția Informatică a Universității Babeș-Bolyai din Cluj și poate fi contactat prin e-mail la adresa: om7280@scs.ubbcluj.ro

Programarea PLĂCII de SUNET

Adrian Monea

Desigur, unul din lucrurile care ne fascinează cel mai mult la calculator este sunetul. De aceea, în această serie de articole ne propunem să abordăm un subiect neglijat până acum în revistele pentru elevi, dar, sperăm noi, de mare interes pentru toți cei care și-au pus problema să genereze sunete de o calitate superioară celor asigurate de PC speaker.

Intrări și ieșiri pentru placa de sunet

Placa de sunet e programată trimițând date regiștrilor săi interni prin cele două porturi de intrare/ieșire ale sale:

$0388 - Port de adresă/stare (Citire / Scriere)
$0389 - Port de date (Numai scriere)

Placa Sound Blaster poate genera muzică FM stereo, care e generată în aceeași manieră. Porturile $0220 si $0221 sunt porturile de adresă/stare respectiv date pentru difuzorul stâng, iar porturile $0222 și $0223 cele pentru difuzorul drept. Porturile $0388 și $0389 vor determina ambele difuzoare să producă sunete.

Placa de sunet are un tablou de 244 de regiștri. Pentru a scrie într-un anumit registru, numărul registrului ($01-$F5) trebuie trimis portului de adresă, iar valoarea dorită portului de date.

Atenție: După scrierea în portul de adresă, programul trebuie să aștepte 12 cicluri de procesor înainte de a scrie valoarea, iar după ce valoarea a fost scrisă, trebuie așteptat 84 de cicluri înainte de a putea face orice altă operație cu placa de sunet. Cea mai precisă metodă de a produce așteptarea este citirea de șase ori a portului de stare după ce am scris în el, și citirea de 35 de ori a portului de stare după scrierea valorii.

Observație:
Regiștrii plăcii de sunet sunt "*read-only*" (pot fi doar citiți).

Portul de adresă funcționează de asemenea și ca port de stare. Pentru a cerceta starea plăcii de sunet, este de ajuns să citim portul $0388.

Octetul de stare a plăcii de sunet

Bit	7	6	5	4	3	2	1	0
Semnificație	Ambele timere	Timerul1	Timerul2	Nefolosiți				

Bitul 7: Egal cu 1 dacă oricare din cele două timere a expirat.
Bitul 6: Egal cu 1 dacă primul timer a expirat
Bitul 5: Egal cu 1 dacă cel de-al doilea timer a expirat.

Regiștrii

Observație:
Regiștrii care nu sunt folosiți nu sunt listați.

Funcțiile regiștrilor plăcii de sunet

Adresă	Funcție
$01	Test LSI/Enable waveform control
$02	Timer 1 data
$03	Timer 2 data
$04	Timer control flags
$08	Speech synthesis mode/Keyboard split note select
$20..$35	Amp Mod/Vibrato/EG type/Key Scaling/Multiple
$40..$55	Key scaling level/Operator output level
$60..$75	Attack Rate/Decay Rate
$80..$95	Sustain Level/Release Rate
$A0..$A8	Frequency (low 8 bits)
$B0..$B8	Key On/Octave/Frequency (high 2 bits)
BD	AM depth/Vibrato depth/Rhythm control
C0..C8	Feedback strength/Connection type
E0..F5	Wave Select

Grupările de câte 22 de regiștri (de exemplu $20..$35, $40..$55 etc.) au o ordine mai ciudată datorită folosirii a doi operatori pentru fiecare voce FM. Tabelul 3 prezintă offseturile pentru fiecare operator în cadrul fiecărui grup de regiștri.

Astfel, adresele octeților de attack/decay pentru canalul 3 sunt 62 pentru primul operator și 65 pentru al doilea. (Adresa celui de-al doilea operator este întotdeauna adresa primului ope-

rator plus 3). Pentru a ilustra mai clar aceasta relație, vom prezenta adresele necesare pentru a controla canalul 5:

Offseturile operatorilor

Canal	1	2	3	4	5	6	7	8	9
Operator 1	00	01	02	08	09	0A	10	11	12
Operator 2	03	04	05	0B	0C	0D	13	14	15

29 - Operator 1 AM/VIB/EG/KSR/Multiplier
2C - Operator 2 AM/VIB/EG/KSR/Multiplier
49 - Operator 1 KSL/Output Level
4C - Operator 2 KSL/Output Level
69 - Operator 1 Attack/Decay
6C - Operator 2 Attack/Decay
89 - Operator 1 Sustain/Release
8C - Operator 2 Sustain/Release
A4 - Frequency (low 8 bits)
B4 - Key On/Octave/Frequency (high 2 bits)
C4 - Feedback/Connection Type
E9 - Operator 1 Waveform
EC - Operator 2 Waveform

Semnificațiile regiștrilor

Registrul $01: Acest octet este folosit în mod normal pentru a testa dispozitivul LSI. În mod normal, toți biții ar trebui să fie 0. Dacă este activat (egal cu 1), bitul 5 permite chipului FM să controleze forma de undă (*waveform*) a fiecărui operator.

Registrul $02: *Timer1 Data*. Dacă Timer 1 este activat, valoarea din acest registru va fi incrementată până când se produce o depășire (*overflow*). Când depășirea se produce, placa de sunet va semnala o întrerupere TIMER (INT 08) și va seta biții 7 și 6 din octetul ei de stare. Valoarea pentru acest timer este incrementată la fiecare 80 de microsecunde.

Registrul $03: *Timer 2 Data*. Dacă Timer 2 este activat, valoarea din acest registru va fi incrementată până când se produce o depășire (*overflow*). Când depășirea se produce, placa de sunet va semnala o întrerupere TIMER (INT 08) și va seta biții 7 și 5 din octetul ei de stare. Valoarea pentru acest timer este incrementată la fiecare 320 de microsecunde.

Registrul $04: *Timer Control Byte*

Bit 7: Resetează steagurile (*flags*) pentru Timer 1 și Timer 2. Dacă acest bit este setat, toți ceilalți biți sunt ignorați.
Bit 6: Maschează Timer 1. Dacă e setat, bitul 0 este ignorat.
Bit 5: Maschează Timer 2. Dacă e setat, bitul 1 este ignorat.
Bit 1: Când nu este setat, Timer 2 nu funcționează. Când este setat, valoarea din octetul $03 este încărcată în Timer 2 și incrementarea începe.
Bit 0: Când nu este setat, Timer 1 nu funcționează. Când este setat, valoarea din octetul $02 este încărcată în Timer 1 și incrementarea începe.

Registrul $08: *CSM Mode/Keyboard split*.

Bit 7: Când este setat, selectează "*composite sine-wave speech synthesis mode*" (toți biții *KEY-ON* trebuie să fie dezactivați). Când nu este setat, selectează modul de muzică FM.
Bit 6: Selectează "*keyboard split point*" (în conjuncție cu datele F-Number).

Regiștrii $20..$35: *Amplitude Modulation/Vibrato/Envelope Generator Type/Keyboard Scaling Rate/Modulator Frequency Multiple*.

Bit 7: Aplică modulare de amplitudine când este setat. Adâncimea AM este controlată de flagul AM-Depth la adresa BD.
Bit 6: Aplică *vibrato* atunci când este setat. Adâncimea vibrației este controlată de flagul Vib-Depth la adresa BD.
Bit 5: Când e setat, nivelul susținut al vocii este menținut până e eliberat. Atunci când nu este setat, sunetul începe să decadă imediat atunci când atinge faza "*sustain*".
Bit 4: *Keyboard scaling rate*. Când este setat, forma sunetului este scurtată atunci când el crește.
Biții 3..0: Acești biți indică in ce armonică va produce sunet (sau modulație) operatorul în relație cu frecvența specificată a vocii:

$0 - o octavă mai jos
$1 - la frecvența specificată a vocii
$2 - o octavă mai sus
$3 - o octavă și o cvintă mai sus
$4 - două octave mai sus
$5 - două octave și o terță majoră mai sus
$6 - două octave și o cvintă mai sus
$7 - două octave și o septimă mai sus
$8 - trei octave mai sus
$9 - trei octave și o secundă majoră mai sus
$A - trei octave și o terță majoră mai sus
$B - "
$C - trei octave și o cvintă mai sus
$D - "
$E - trei octave și o septimă majoră mai sus
$F - "

Regiștrii $40..$55: *Level Key Scaling/Total Level*.

Observație

Valorile din tabelul de mai sus sunt date în decibeli (Db).
Biții 7..6: Provoacă scăderea nivelelor de ieșire pe măsură ce frecvența crește:

00 - nici o schimbare
10 - 1.5 dB/8ve
01 - 3 dB/8ve
11 - 6 dB/8ve

Biții 5..0: Controlează nivelul total de ieșire a operatorului. Când toți biții sunt dezactivați nivelul este cel mai ridicat. Când toți biții sunt selectați, nivelul e cel mai scăzut.

Regiștrii $60 - $75: *Attack Rate/Decay Rate*.

Semnificația registrilor

Bit	7	6	5	4	3	2	1	0
	Semnificație							
Registrul $01	Nefolosiți		Waveform set		Nefolosiți			
Registrul $04	IRQ RST	T1 MSK	T2 MSK		Nefolosiți		T2 CTL	T1 CTL
Registrul $08	CSM Sel	Key Spl			Nefolosiți			
Regiștrii $20...$35	Amp Mod	Vib	EG Type	KSR	Modulator Frequency Multiple			
Regiștrii $40...$55	Scaling level		Total Level					
			24	12	6	3	1.5	0.75
Regiștrii $60...$75	Attack Rate				Decay Rate			
Regiștrii $80...$95	Sustain Level				Release Rate			
	24	12	6	3				
Regiștrii $A0...$A8	Nefolosiți		Key On		Octave		F-Number (Cei mai semnificativi biți)	
Regiștrii $C0...$C8	Nefolosiți				Feedback			Alg
Registrul $BD	Dep	Vib Dep	Rhy Ena	BD	SD	TOM	Top Cym	HH
Regiștrii $E0...$F5	Nefolosiți						Waveform Select	

Biții 7..4: *Attack rate*. 0 este cel mai încet, iar F este cel mai rapid.
Biții 3..0: *Decay rate*. 0 este cel mai încet, iar F este cel mai rapid.

Regiștrii $80..$95: *Sustain Level/Release Rate*
Biții 7..4: - *Sustain Lebel*. 0 este cel mai tare, iar $F este cel mai încet.
Biții 3..0: - *Release Rate*. 0 este cel mai încet, iar $F este cel mai rapid.

Regiștrii $A0..$B8: *Octave/F-Number/Key-On*.

Regiștrii $B0 .. $B8
Bitul 5: Canalul este activat când bitul este setat și dezactivat când bitul este 0.

Biții 4..2: Octava(0-7). 0 este cea mai joasă, iar 7 este cea mai înaltă.
Biții 1..0: Cei mai semnificativi biți ai lui F-Number.

În octava a patra, *F-Number* are valoarea scării cromatice, iar frecvențele lor corespunzătoare sunt cuprinse în tabelul:

F-Number	Frecvență	Notă
$16B	277.2	C#
$181	293.7	D
$198	311.1	D#
$1B0	329.6	E
$1CA	349.2	F
$1E5	370.0	F#
$202	392.0	G
$220	415.3	G#
$241	440.0	A
$263	446.2	A#
$287	493.9	B
$2AE	523.3	C

Regiștrii $C0..$C8: *Feedback/Algorithm*.
Biții 3..1: Puterea feedbackului. Dacă toți trei biții sunt se-

taţi pe 0, feedbackul nu are loc. Cu valori între 1 şi 7, operatorul 1 trimite o cantitate din ieşirea sa înapoi la el. 1 este cea mai mică doză de feedback, iar 7 este cea mai mare.

Bitul 0: Dacă nu este setat, operatorul 1 modulează operatorul 2. În acest caz, operatorul 2 este singurul care produce sunet. Dacă este setat, ambii operatori produc sunet în mod direct. Sunetele complexe sunt mai uşor produse dacă bitul nu este setat.

Registrul $BD: *Amplitude Modulation Depth/Vibrato Depth/Rhythm*.

Bitul 7: Dacă este setat, adâncimea AM este 4.8 dB. Dacă nu este setat, AM depth este 1 dB.

Bitul 6: Dacă este setat, adâncimea Vibrato este 14. Dacă nu este setat, adâncimea este 7.

Bitul 5: Dacă este setat, Rhytm este activat (există 6 voci melodice). Dacă este dezactivat, Rhytm este dezactivat (există 9 voci melodice).

Bitul 4: Bass drum activat/dezactivat.

Bitul 3: Snare drum activat/dezactivat.

Bitul 2: Tom activat/dezactivat.

Bitul 1: Cymbal activat/dezactivat.

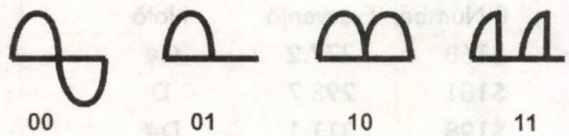
Bitul 0: Hi Hat activat/dezactivat.

Observaţie:

Regiştrii KEY-ON pentru canalele $06, $07 şi $08 trebuie să fie dezactivaţi pentru a folosi secţiunea de ritm. Alţi parametri ca *attack/decay/sustain/release* trebuie de asemenea setaţi corespunzător.

Regiştrii $E0..$F5: *Waveform select*.

Biţii 1..0: Când bitul 5 al adresei $01 este setat, unda de ieşire va fi distorsionată corespunzător undei indicate de aceşti doi biţi. În continuare vom prezenta o diagramă aproximativă a undei în funcţie de valorile posibile.



Detectarea plăcii de sunet

Conform manualului oficial, metoda corectă de verificare a plăcii de sunet este următoarea:

1) Se resetează ambele timere scriind $60 în registrul $4.
2) Se activează întreruperile scriind $80 în registrul $4.

Observaţie:

Această operaţie trebuie făcută separat de pasul 1.)

3) Se citeşte registrul de stare (portul $388). Se păstrează rezultatul.
4) Se scrie $FF în registrul $2 (Timer 1).
5) Se porneşte timerul 1 scriind valoarea $21 în registrul $4.
6) Se provoacă o pauză de cel puţin 80 microsecunde.
7) Se citeşte registrul de stare (portul $388). Se păstrează rezultatul.
8) Se resetează ambele timere şi întreruperile (vezi paşii 1 şi 2).
9) Se testează rezultatele păstrate la paşii 3 şi 7 făcând o operaţie ŞI LOGIC pe biţi (AND) cu $E0. Rezultatul de la pasul 3 ar trebui să fie $0, iar rezultatul de la pasul 7 ar trebui să fie $C0. Dacă ambele rezultate sunt corecte, o placă Sound Blaster (sau compatibilă) se află instalată în calculator.

Producerea unui sunet

În numărul următor vom prezenta câteva exemple mai complexe de programare a plăcii de sunet. Până atunci, pentru cei care nu mai au răbdare, vom prezenta valorile regiştrilor, necesare pentru a genera un sunet simplu.

În primul rând, resetaţi toţi regiştrii iniţializându-i cu 0. Aceasta este o metodă rapidă de resetare a plăcii de sunet, care nu apare în documentaţia oficială, însă funcţionează. Dacă doriţi să folosiţi unde diferite, bitul 5 al regiştrului $1 trebuie setat pe 1. (Această resetare trebuie făcută doar o dată, la începutul programului şi doar opţional când programul se termină, pentru a fi siguri că programul nu lasă nici o urmă atunci când se termină).

Apoi încărcaţi următorii regiştri cu valorile indicate în tabelul:

Registru	Valoare	Descriere
$20	$01	Setează multiplul modulatorului pe 1.
$40	$10	Setează nivelul modulatorului la aproximativ 40 dB.
$60	$F0	Modulator attack: rapid, decay: lung.
$80	$77	Modulator sustain: medium, release: medium.
$A0	$98	Setează LSB-ul frecvenţei vocii (va fi D#).
$23	$01	Setează multiplul pentru purtător pe 1.
$43	$0	Setează volumul pe maxim (aproximativ 47 dB).
$63	$F0	Carrier attack: rapid, decay: lung.
$83	$77	Carrier sustain: mediu, release: mediu.
$B0	$31	Activează vocea. Setează MSB-ul pentru octavă şi frecvenţă

Pentru a dezactiva vocea, setaţi registrul $B0 la $11 (sau la orice valoare al cărei bit 5 este 0). Bineînţeles, în general este preferabil să se producă o întârziere înainte de această operaţie.

Bibliografie:

1. Mark Feldman - *PCGPE* (PC Games Programming Encyclopedia, format electronic)
2. Creative Labs Corporation - *Sound Blaster Resource Kit*

Adrian Monea este student în anul I la Facultatea de Automatică şi Calculatoare, Universitatea Tehnică Cluj, poate fi contactat prin intermediul redacţiei.

Acces și VIZIBILITATE

Horia Georgescu

Vom studia în continuare drepturile de acces la clase, metode, câmpuri și variabile. Toate aceste entități sunt identificate prin numele lor. Problema care se pune este de a stabili din ce loc putem face referire la aceste nume (cu semnificația dorită), adică de a preciza *domeniul de vizibilitate* al acestora. Cu acest prilej vom face o discuție mai amănunțită referitoare la modificatori. Discuția va fi restrânsă la stadiul actual al prezentării, care face abstracție de facilitatea de extindere a claselor, și va fi reluată ulterior.

Variabile locale și etichete

Prin *bloc* înțelegem o secvență de zero sau mai multe instrucțiuni, cuprinsă între acolade. Un bloc *este o instrucțiune* și de aceea poate fi folosit în orice loc unde poate să apară o instrucțiune.

De exemplu orice metodă și orice constructor sunt formate dintr-un antet urmat de un bloc. În schimb ceea ce urmează după modificatorii și numele unei clase nu constituie un bloc, deși apar acoladele deschisă și închisă: conținutul unei clase nu este o succesiune de instrucțiuni.

Orice bloc poate conține la rândul său un alt bloc, putând astfel lua naștere o structură imbricată de blocuri.

Prin *variabilă locală* înțelegem o variabilă declarată în interiorul unui bloc, inclusiv într-o instrucțiune `for`.

Domeniul de vizibilitate al unei variabile locale este constituit din partea din blocul în care a fost declarată, ce urmează declarării. Prin urmare o variabilă locală nu poate fi referită înainte de a fi declarată. Variabilele locale există numai pe perioada în care blocul în care sunt declarate este în curs de executare.

Nu putem folosi facilitatea de imbricare a blocurilor pentru a redeclara o variabilă locală. O variabilă locală nu poate fi redefinită nici în blocul în care a fost declarată, nici într-un bloc inclus în acesta.

Regulile care guvernează domeniul de vizibilitate al etichetelor sunt aceleași ca pentru variabilele locale.

Parametrii metodelor și constructorilor

Numele parametrilor dintr-o metodă sau dintr-un constructor sunt vizibile doar în interiorul acestora, adică în blocul care urmează antetului metodei sau constructorului. Este însă posibil ca într-un bloc dintr-o metodă sau dintr-un constructor să redeclarăm numele unui parame-

Elemente

Lucrul cu șiruri în Java

Șirurile din Java sunt obiecte standard. Orice șir este un obiect al clasei `String`, ce extinde clasa "rădăcină" `Object`.

Cunoaștem deja câțiva operatori pentru lucrul cu șiruri:
- operatorul `+`, folosit pentru concatenare:
- operatorul `+=`, ce creează un nou șir prin concatenarea șirului din stânga operatorului cu șirul din dreapta sa.

Este important de reținut că un (obiect de tip) șir nu poate fi modificat (este *"read-only"*) dar, prin utilizarea metodelor clasei `String`, plecând de la el, putem obține noi șiruri.

Clasa `String` conține constructorii:
`public String()` și `public String(String` *șir* `)`
care permit să construim un nou șir inițializat cu șirul vid, respectiv cu *șir*.

Exemplu:

```
...
String s1,s2; ...
s1=new String(); // echivalent cu s1="";
s2=new String("abc");
                  //echivalent cu s2+"abc";
```

Toate metodele clasei `String` au modificatorul de acces `public`. Trecem la prezentarea câtorva dintre ele.
Metodele esențiale ale clasei `String` sunt:
- `length()`: întoarce lungimea șirului care a invocat metoda:

- `charAt(int i)`: întoarce caracterul de pe poziția `i` din șirul care a invocat metoda.

Să observăm diferențele față de lucrul cu tablouri. Dacă `s` este un șir, iar `a` un tablou, atunci:

- `s.length()` este lungimea șirului, iar `a.length` este lungimea tabloului;
- `s.charAt(i)` este caracterul de pe poziția `i` din șir, iar `a[i]` este elementul de pe poziția `i` din tablou.

Invocarea metodei `charAt` cu un indice în afara șirului lansează excepția: `IndexOutOfBoundsException`. Această excepție mai poate fi lansată și de alte metode prezentate în continuare (identificarea lor se face cu ușurință).

Metode pentru aflarea unui indice din șir

Următoarele metode ale clasei `String` întorc o poziție cu o anumită proprietate din șirul `s` prin care au fost invocate (dacă o astfel de poziție nu există, este întoarsă valoarea `-1`).

Metodă — Întoarce indicele din șir de pe...

`indexOf(char ch)`
prima poziție pe care apare `ch`

`indexOf(char ch, int p)`
prima poziție, mai mare sau egală cu `p`, unde apare `ch`

`indexOf(String sub)`
prima poziție începând de la care apare șirul `sub`

`indexOf(String sub, int p)`
prima poziție, mai mare sau egală cu `p`, începând de la care apare șirul `sub`

`lastIndexof(char ch)`
ultima poziție pe care apare `ch`

`lastIndexof(char ch, int u)`
ultima poziție, cel mult egală cu `u`, pe care apare `ch`

`lastIndexOf(String sub)`
ultima poziție începând de la care apare șirul `sub`

`lastIndexof(String sub, int u)`
ultima poziție, cel mult egală cu `u`, începând de la care apare șirul `sub`

Exemplu

Următorul program citește două șiruri și determină numărul de apariții al celui de al doilea în primul. Șirurile sunt citite de pe linia de comandă, la executare.

```
class nr_apar {
 static int nrapar(String s,String subs){
  int nr=0,indice,poz=0;
  indice=s.indexOf(subs,poz);
  while (indice>=0 && indice<s.length()){
   nr++; poz++;
   indice=s.indexOf(subs,poz);
  }
  return nr;
 }
```

tru; în acest caz, în partea din bloc ce urmează redeclarării, numele respectiv este considerat ca având semnificația dată de redeclarare.

Cu alte cuvinte, domeniul de vizibilitate al unui parametru este constituit din corpul metodei sau constructorului în care apare, mai puțin blocurile (evident disjuncte) în care este redeclarat.

Vizibilitatea în interiorul unei clase

Știm că o clasă este constituită din câmpuri, constructori (folosiți la crearea de obiecte) și metode (folosite la invocarea lor). Din orice constructor și din orice metodă putem să ne referim la orice câmp sau metodă a clasei. De asemenea dintr-un constructor putem apela alt constructor prin **this** urmat, între paranteze, de o listă de parametri ce respectă signatura noului constructor.

Ordinea în care apar câmpurile și metodele în cadrul clasei nu este relevantă (de exemplu nu contează dacă un câmp este declarat la începutul sau sfârșitul clasei).

Totuși, dacă numele unui câmp este redeclarat într-o metodă (este folosit ca nume al unui parametru sau declarat într-un bloc), atunci declararea cea mai interioară este cea care primează când se face o referire la acel nume (identificator). Putem spune că un parametru poate *ascunde* un câmp, iar o variabilă locală poate *ascunde* un parametru și/sau un câmp.

După cum vom vedea în continuare, numele unei metode nu poate fi ascuns.

Modificatorul final

Modificatorul **final** poate fi atașat atât variabilelor (locale sau câmpuri), cât și metodelor și claselor.

În cazul variabilelor locale, variabilei i se atribuie valoare la declarare sau înainte de a fi folosită; variabila nu poate primi de două ori valoare.

Unui câmp final trebuie să i se atribuie valoare fie prin inițializare, fie prin toți constructorii. Când câmpul este static, există deci posibilitatea de a i se atribui o valoare printr-un bloc de inițializare static:

Exemplu

```
class Clasa {
  static final int x; static { x=-1;}

  public static void met() {
    System.out.println(x);
    final int x; x=3; System.out.println(x);
  }
}

public class temp {
  public static void main (String[] s) {
    Clasa Ob = new Clasa(); Ob.met();
  }
}
```

cu precizarea că era mai natural ca invocarea metodei statice `met` să se fi făcut prin `Clasa.met();` adică fără a mai crea o instanță `Ob` a clasei.

Menționăm însă că în general modificatorul **`final`** este folosit, însoțit de **`static`**, pentru a trata un câmp ca o "constantă cu nume".

Efectul modificatorului **`final`** asupra metodelor și claselor va fi prezentat atunci când vom vorbi despre extinderea claselor.

Modificatorul static

Modificatorul **`static`** poate fi folosit la declararea câmpurilor și metodelor unei clase; cuvântul cheie **`static`** mai este folosit și pentru a anunța un bloc de inițializare **`static`**.

O variabilă locală nu poate fi declarată cu **`static`**.

Fie `c` un câmp declarat cu **`static`** în clasa `Clasa`. Această declarare are două consecințe:

- câmpul `c` este comun tuturor obiectelor de tipul `Clasa`;
- referirea la câmp din exteriorul clasei se face fie conform regulii generale (prin crearea și utilizarea unei instanțe a clasei `Clasa`) fie direct, prin precalificarea câmpului cu numele clasei (`Clasa.c`).

Câmpurile statice sunt inițializate înainte de a fi folosite și înainte ca vreo metodă a clasei respective să fie executată.

Fie acum `met` o metodă declarată cu modificatorul **`static`** în clasa `Clasa`; ea se numește *metodă statică* sau *metodă de clasă*. Invocarea ei se poate face fie conform regulii generale (prin crearea și utilizarea unei instanțe a clasei `Clasa`) fie direct, prin precalificarea metodei cu numele clasei: `Clasa.met(...);` de aceea în metodă nu poate fi folosită referința **`this`**.

O metodă statică poate accesa numai câmpurile și metodele statice ale clasei. Metodele statice (de clasă) sunt folosite de obicei pentru a efectua prelucrări asupra câmpurilor statice ale clasei. Putem "ocoli" însă această regulă, transmițând (ca parametru) metodei o referință la un obiect, dar acest mod de lucru apare ca nenatural.

Lucrul cu blocuri de inițializare statice a fost descris într-un paragraf precedent.

Modificatorii de acces

Toate câmpurile și metodele unei clase sunt accesibile (pot fi referite) din interiorul clasei.

Rolul modificatorilor **`public`**, **`protected`**, **`private`** și cel implicit (numit uneori *package*) referitor la accesul din exteriorul unei clase la membrii săi este:

Modificator	*Membrii clasei sunt accesibili*
`public`	- de oriunde clasa este accesibilă
implicit (*package*)	- din cod din același pachet
`protected`	- din cod din același pachet
`private`	- numai din interiorul clasei

```
 public static void main(String[] s) {
  System.out.println( nrapar(s[0],s[1]));
 }
}
```

Compararea șirurilor

Clasa `String` prevede metode pentru compararea conținutului șirului care invocă metoda conținutul unui șir transmis ca parametru. Vom presupune că șirul care invocă metodele este `s`. Compararea se referă la șirurile în totalitatea lor sau la subsecvențe ale lor.
Compararea se face conform ordinii lexicografice; compararea a două caractere constă în compararea numerelor lor de ordine în setul de caractere Unicode.

`boolean` `equals(String s1)`
: întoarce `true` dacă șirurile coincid;

`int` `CompareTo(String s1)`
: întoarce o valoare negativă (dacă `s<s1`), zero (dacă `s=s1`) sau o valoare pozitivă (dacă `s>s1`);

`boolean` `regionMatches(int p,String s1, int p2,int lung)`
: compară secvențele de lungime `lung` ce încep în `s` de pe poziția `p`, respectiv în `s1` de pe poziția `p1`; este întoarsă valoarea `true` dacă și numai dacă aceste secvențe coincid;

`boolean` `startsWith(String pre)`
: întoarce `true` dacă șirul `pre` este prefix al lui `s` și `false` altfel;

`boolean` `endsWith(String suf)`
: întoarce `true` dacă șirul `suf` este sufix al lui `s` și `false` altfel.

Să remarcăm că metoda `equals` nu poate fi suplinită de operatorul `==` deoarece prin `s==s1` se compară două referințe și nu conținutul lor!

Determinarea unui subșir cu o anumită proprietate

Metodele clasei `String` ce se încadrează aici sunt următoarele (vom presupune din nou că șirul care invocă metoda este notat prin `s`):

`String substring(int p)`
: este întors subșirul lui `s` ce începe de pe poziția `p`;

`String substring(int p, int u)`
: este întors subșirul lui `s` ce începe de pe poziția `p` și se termină pe poziția `u-1`;

`String replace(char old, char new)`
: este întors șirul obținut din `s` prin înlocuirea tuturor caracterelor `old` prin caracterul `new`;

`String toUpperCase()`
: fiecare caracter este înlocuit cu echivalentul său majusculă, dacă acesta există;

`String toLowerCase()`
: fiecare "literă mică" este înlocuită cu "literă mare",

dacă această corespondență există;

`String trim()`

întoarce șirul obținut din `s` prin eliminarea spațiilor albe de la începutul și sfârșitul său;

`String concat(String s1)`

întoarce șirul `s+s1`.

Șiruri și tablouri de caractere

Clasa `String` prevede metode pentru transformarea șirurilor în tablouri de caractere, precum și pentru transformarea inversă:

Metoda statică `toCharArray()` invocată de un șir `s` întoarce un tablou de caractere de aceeași lungime și cu aceleași elemente.

Metoda statică `copyValueOf` are două forme:

```
String copyValueOf(char[] a)
String copyValueOf(char[] a, int p,
                                int lung)
```

și întoarce șirul corespunzător tabloului `a` de caractere, respectiv numai acea secvență din tablou ce începe pe poziția `p` și are lungimea `lung`.

De exemplu, pentru a determina șirul obținut din șirul `s` prin considerarea elementelor sale în ordine inversă, putem scrie următoarea metodă:

```
public static String invers(String s) {
  char c;
  char[] a=s.toCharArray();
  for (int p=0,u=a.length-1; p<u; p++,u--)
  {
    c=a[p]; a[p]=a[u]; a[u]=c;
  }
  return String.copyValueOf(a);
}
```

O altă metodă pentru compararea șirurilor

Metoda `String intern()` întoarce un șir având același conținut ca cel al șirului care invocă metoda, având în plus următoarea proprietate: *pentru orice două șiruri cu același conținut, valoarea întoarsă de metoda* `intern` *este aceeași*. Drept urmare putem verifica dacă două șiruri au același conținut prin:

`s1.intern()==s2.intern()` *care este mai rapidă decât:* `s1.equals(s2)`.

Exemplu

De la intrare sunt citite cel mult 100 de șiruri de caractere delimitate de ghilimele. Vrem să reținem doar pe cele distincte, urmând ca în final să le listăm.

```
import java.io.*;
class sirunic {
  static String vect[] = new String[100];
```

Reamintim că modificatorii de mai sus joacă un rol nu numai în privința accesului, dar și pentru facilitatea de extindere a claselor, ceea ce diferențiază modificatorul **protected** și cel implicit. Evident, vom reveni asupra acestor modificatori când vom studia extinderea claselor.

Determinarea semnificației unui identificator

Pentru un identificator folosit ca nume de variabilă (locală sau câmp), metodă sau clasă, semnificația sa se obține la prima căutare încununată de succes din următorul șir de căutări succesive:

1) variabilele locale declarate într-un bloc sau un ciclu **for**;
2) numai dacă suntem în interiorul unei metode sau a unui constructor: parametrii metodei sau constructorului;
3) elemente rezultate prin extinderea claselor (vom face precizări la momentul respectiv);
4) tipuri importate explicit;
5) tipuri declarate în același pachet;
6) tipuri importate implicit;
7) pachete disponibile pe sistemul gazdă.

Mai mult, se face distincție clară între tipurile de identificatori: nume de variabile (locale), de pachete, de clase, de metode și de câmpuri; iată de ce numele unei metode nu poate fi ascuns. Aceasta permite să folosim un același identificator pentru a numi una sau mai multe astfel de entități. De exemplu, următoarea clasă este corectă din punct de vedere sintactic, compilatorul nesemnalând vreo eroare:

```
class obsesie {
  obsesie obsesie; obsesie()
  { obsesie = new obsesie(); }
  obsesie obsesie (obsesie obsesie) {
    obsesie:
    while (obsesie != null)
      if (obsesie.obsesie(obsesie)==obsesie)
        break obsesie;
  return obsesie;
  }
}
```

Nu recomandăm să se abuzeze astfel de distincția menționată. Clasa de mai sus poate fi scrisă sub următoarea formă mai clară și "neobsesivă":

```
class obsesie {
  obsesie C;
  obsesie() { C = new obsesie(); }
  obsesie met(obsesie param){
    et:
    while (param!=null) {
      if (param.met(param)==param) break et;
    }
    return param;
  }
}
```

```
  static int lung;
  static void adaug(String s) {
    String martor = s.intern();
    for (int i=0; i<lung; i++)
      if (vect[i]==martor) return;
    vect[lung++] = martor; return;
  }

  public static void main (String[] str)
                          throws IOException {
    StreamTokenizer f=new
    StreamTokenizer(System.in);
    f.nextToken();
    while (f.ttype=='\"') {
      adaug(f.sval); f.nextToken();
    }
    for (int i=0; i<lung; i++)
      System.out.println(vect[i]);
  }
}
```

Este important de notat că programul funcționează corect deoarece se lucrează doar cu șiruri obținute ca efect al invocării metodei `intern`.

Conversii

În multe cazuri suntem puși în situația de a converti valori numerice în șiruri sau de a lucra cu o valoare numerică memorată într-un șir.
Șirul ce corespunde unei valori numerice sau booleene `x` se obține simplu prin: `String.valueOf(x)`
Fie acum `s` un șir ce conține un literal boolean sau numeric `x`. Atunci x se obține prin:

`new Boolean(s).booleanValue()`, pentru tipul `boolean`;
`new Float(s).floatValue()`, pentru tipul `float`;
`new Double(s).doubleValue()`, pentru tipul `double`;
`Integer.parseInt(s,b)`, pentru tipul `int`;
`Long.parseLong(s,b)`, pentru tipul `long`.
cu mențiunea că atunci când conținutul șirului nu are forma așteptată, este lansată excepția `NumberFormatException`.

Cele de mai sus nu se aplică pentru valori scrise în octal sau hexazecimal (cu prefixele `0`, respectiv `0x`).
Metoda `valueOf` a clasei `String` mai poate fi invocată și pentru un caracter; va fi întors șirul format din acel unic caracter.

Prof. dr. Horia Georgescu este cadru didactic al Facultății de Matematică a Universității din București. E-mail: hg@colegiu.cs.unibuc.ro

Premiul lunii

Începând din această lună, revista noastră va încerca să ofere în fiecare număr câte un *PREMIU DE FIDELITATE* unui cititor ales prin tragere la sorți dintre *ABONAȚI*.

Și tu ai șansa să câștigi *PREMIUL LUNII*. Condiția este simplă: *ABONEAZĂ-TE !*

Premiul lunii noiembrie este oferit de sponsorul nostru, *MICROSOFT ROMANIA*...

... și a fost câștigat de către

RADU CUCU **din Caransebeș.**

Pagini WEB cu HTML 3.2 (II)

Mihai Scorțaru

Cele mai importante și interesante elemente ale unei pagini Web sunt imaginile, 99,9% din paginile Web existente conținând imagini. Dar timpul trece... și imaginea nu se încarcă... Cum să evităm ca vizitatorul paginii noastre să nu apese butonul STOP, pierzându-și răbdarea?

Amplasarea unei imagini într-o pagină Web

Deși au o importanță mare, imaginile conținute în paginile Web constituie și o mare problemă din cauza timpului relativ mare necesar încărcării lor în timpul navigării pe Internet. Majoritatea browserelor conțin o opțiune de tipul *Auto Load Images*. Prin deselectarea acestei opțiuni browserul va fi informat că nu se dorește încărcarea automată a imaginilor în același timp cu textul. În locul unei imagini va apărea un icon, însoțit eventual de un text. Dacă se dorește încărcarea unei imagini trebuie selectat icon-ul corespunzător. Opțiunea de tipul *Auto Load Images* diferă de la un browser la altul și de la o versiune la alta. De exemplu, în Internet Explorer 4.0, opțiunea corespunzătoare este:

View | Internet Options... | Advanced | Multimedia | Show Images.

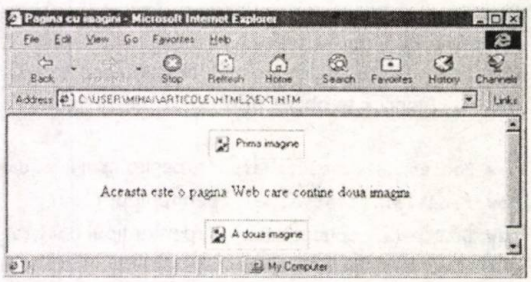
În Internet Explorer 4.0. icon-ul care apare de obicei este:

Tag-ul HTML cu ajutorul căruia se afișează o imagine este <IMG>. Cei mai importanți parametri ai săi sunt: **SRC** și **ALT**. Cu ajutorul parametrului SRC se precizează fișierul în care este conținută imaginea, iar cu ajutorul parametrului ALT se precizează un text alternativ care va apărea în cazul în care imaginea nu poate fi încărcată din diferite motive cum ar fi: formatul nu este suportat de browser, fișierul respectiv lipsește sau este corupt, este deselectată opțiunea de tipul *Auto Load Images* etc.

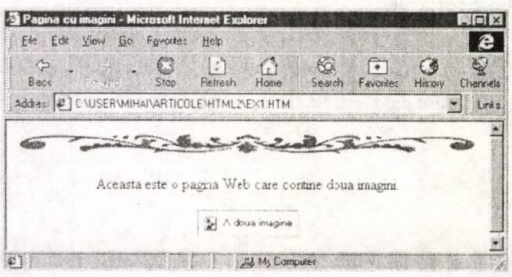
Să considerăm fișierul HTML:

```
<html>
<head>
<title>Pagina cu imagini</title>
</head>
<body>
<div align = "center">
<img src="img1.gif" alt="Prima imagine">
<br><br>Aceasta este o pagina Web care contine
doua imagini.<br><br>
<img src="img2.gif" alt="A doua imagine">
</div>
</body>
</html>
```

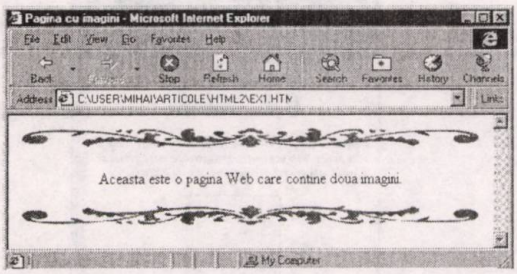
În cazul în care imaginile nu pot fi încărcate sau nu este dorită încărcarea lor, pagina Web corespunzătoare va arăta astfel:



Pentru a cere afișarea unei imagini, comenzile diferă de la un browser la altul. Pentru *Netscape Navigator* nu trebuie decât să efectuați un left-click pe icon-ul care a fost afișat în locul imaginii. Pentru *Internet Explorer 4.0* trebuie efectuat un right-click pe icon-ul corespunzător și din meniul pop-up care va apărea trebuie selectată comanda *Show picture*. Dacă vom cere încărcarea primei imagini și această cerință va putea fi îndeplinită, va fi afișată următoarea pagină:



Se observă că a doua imagine nu este încărcată automat atunci când se cere încărcarea primei imagini. Pentru a vedea și cealaltă imagine trebuie să efectuăm aceleași operații ca și pentru prima. În cazul în care a doua imagine este disponibilă ea va fi încărcată, iar pagina Web corespunzătoare va arăta astfel:



Trebuie precizat că, în cazul în care cele două imagini ar fi fost identice (atributul SRC al tag-ului IMG ar fi conținut același fișier) atunci, dacă s-ar fi cerut încărcarea uneia din cele două imagini, ar fi fost afișate amândouă.

Alinierea imaginilor

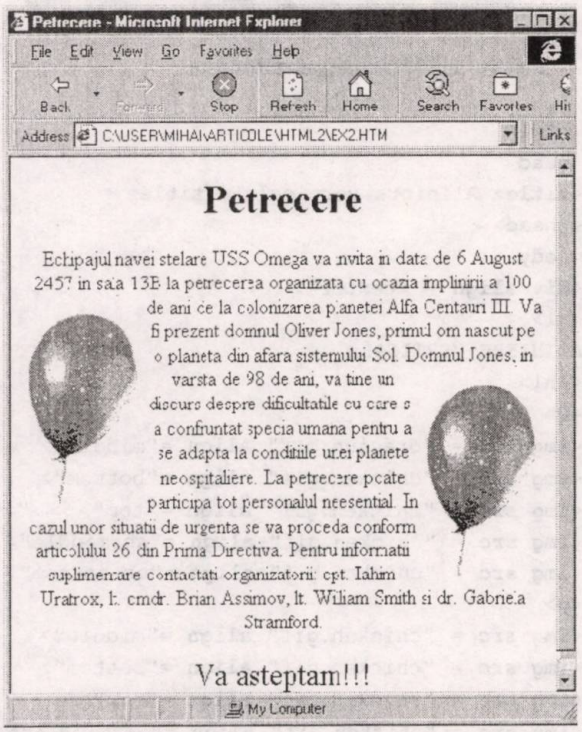
Tag-ul html <IMG> mai are un parametru important și anume ALIGN cu ajutorul căruia pot fi aliniate imaginile. O imagine poate fi aliniată atât orizontal, cât și vertical.

Pentru alinierea orizontală, valorile parametrului **ALIGN** pot fi left, right sau center. Pentru alinierea verticală valorile pot fi: top, bottom, absbottom, middle sau absmiddle.

În cazul alinierii orizontale imaginea va fi aliniată la dreapta, la stânga sau va fi centrată. Trebuie observat faptul că parametrul **ALIGN** al comenzii **IMG** diferă de același parametru al comenzii **DIV**. De exemplu <IMG ALIGN = "right"> va alinia o imagine la dreapta și textul care urmează va "înfășura" imaginea. Acest fapt este ilustrat în următorul exemplu:

```
<html>
<head>
<title> Petrecere </title>
</head>
<body>
<div align="center">
<h1> Petrecere </h1>
   Echipajul navei stelare USS Omega va
invita in data de 6 August 2457 in sala 13B
la petrecerea organizata cu
<img src = "balon.gif" alt = "balon" align =
"left">
ocazia implinirii a 100 de ani de la
colonizarea planetei Alfa Centauri III. Va
fi prezent domnul Oliver Jones, primul om
nascut pe o planeta din afara sistemului
Sol.
<img src = "balon.gif" alt = "balon" align =
"right">
Domnul Jones, in varsta de 98 de ani, va
tine un discurs despre dificultatile cu care
s-a confruntat specia umana pentru a se
adapta la conditiile unei planete
neospitaliere. La petrecere poate participa
tot personalul neesential. In cazul unor
situatii de urgenta se va proceda conform
articolului 26 din Prima Directiva. Pentru
informatii suplimentare contactati
organizatorii: cpt. Iahim Uratrox, lt. cmdr.
Brian Assimov, lt. William Smith si dr.
Gabriela  Stramford.
<p>
<basefont size=5>
Va asteptam!!!
</div>
</body>
</html>
```

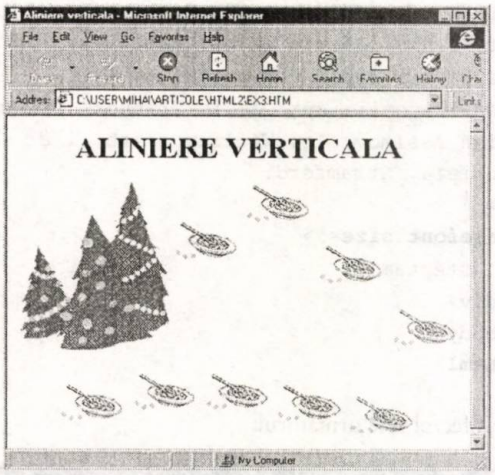
Efectul este următorul:



Valorile posibile ale parametrului **ALIGN** pentru alinierea verticală sunt: top, bottom, absbottom, middle și absmiddle. Dacă se folosește valoarea top marginea superioară a imaginii va fi aliniată cu marginea superioară a celei mai "înalte" imagini sau litere de pe linia pe care apare imaginea. Pentru a alinia marginea inferioară a imaginii cu marginea inferioară a textului folosiți valoarea bottom. Dacă doriți ca marginea inferioară a imaginii să fie aliniată

cu marginea inferioară a celei mai "de jos" imagini, trebuie folosită valoarea `absbottom`. Valoarea `middle` este folosită pentru a alinia mijlocul imaginii cu mijlocul textului. Pentru a alinia mijlocul imaginii cu centrul absolut a tot ceea ce apare pe linie, folosiți valoarea `absmiddle`.

Pe pagina următoare veți vedea o pagină Web care ilustrează folosirea alinierii verticale:



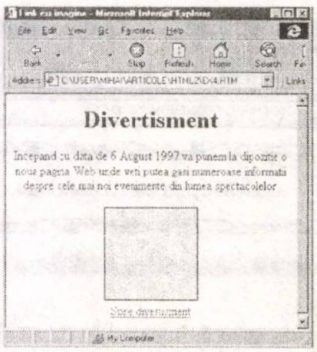
Fișierul HTML corespunzător este:

```
<html>
<head>
<title> Aliniere verticala </title>
</head>
<body>
<div align = "center">
<h1>
ALINIERE VERTICALA
</h1>
<P>
<img src = "craciun.gif" align ="middle">
<img src = "chicken.gif" align ="bottom">
<img src = "chicken.gif" align ="top">
<img src = "chicken.gif" align ="absmiddle">
<img src = "chicken.gif" align ="absbottom">
<p>
<img src = "chicken.gif" align ="middle">
<img src = "chicken.gif" align ="bottom">
<img src = "chicken.gif" align ="top">
<img src = "chicken.gif" align ="absmiddle">
<img src = "chicken.gif" align ="absbottom">
</div>
</body>
</html>
```

Se observă că, datorită faptului că prima imagine din primul rând este foarte mare, imaginile mici din primul rând au "diferențe de nivel" mult mai mari decât cele din al doilea rând.

Imagini care sunt link-uri

Dacă se dorește ca un link să conțină și o imagine grafică acest lucru poate fi realizat prin includerea unei comenzi `<IMG>` între opening tag-ul și closing tag-ul comenzii `<A>`. Un astfel de link poate fi văzut în imaginea următoare:



Indiferent dacă veți apăsa pe baloane sau pe textul de sub ele browser-ul va căuta fișierul `DIVERTISMENT.HTML` și va încerca să afișeze pagina Web corespunzătoare. Fișierul HTML care are acest efect este următorul:

```
<html>
<head>
<title> Link cu imagine </title>
</head>
<body>
<div align = "center">
<h1> Divertisment </h1>
Incepand cu data de 6 August 1997 va punem
la dipozitie o noua  pagina Web unde veti
putea gasi numeroase informatii despre cele
mai noi evenimente din lumea spectacolelor.

<p>
<a href = "divertisment.html">
<img src = "baloane.jpg"> <br>
Spre divertisment
</a>
</body>
</html>
```

Formate de imagini suportate de browserele HTML

Dacă un browser respectă standardul HTML atunci el va recunoaște cu siguranță formatele GIF și JPEG. Unele browsere pot recunoaște și alte formate cum ar fi BitMap, PCX etc., dar nu se recomandă folosirea unor imagini în aceste formate deoarece majoritatea browserelor nu le vor putea afișa.

Micșorarea timpului necesar pentru încărcarea unei pagini

În majoritatea cazurilor, alături de fișierul `.HTM` sau `.HTML` sunt încărcate multe alte fișiere care conțin diferite infor-

mații. Cele mai frecvente astfel de fișiere sunt cele care conțin imagini. Datorită numărului mare de octeți care trebuie încărcați, *intervalul de timp dintre momentul în care este comandată afișarea unei imagini și momentul în care ea este afișată complet, este destul de mare*. Timpul necesar încărcării unei pagini este foarte important, deoarece un navigator pe Internet nu va aștepta prea mult ca o pagină să se încarce și după un interval de timp mai mic sau mai mare va renunța la vizitarea acestei pagini. Bineînțeles, nici un proiectant de pagini Web nu dorește ca butonul STOP al browserului să fie apăsat în timp ce pagina sa este încărcată.

O practică des întâlnită este micșorarea numărului de culori care apar în imagine la 256 sau 16. Aceasta va reduce considerabil dimensiunea fișierului care conține imaginea și acel fișier va putea fi încărcat mult mai repede.

O altă modalitate este folosirea unor formate GIF sau JPEG speciale numite Interlaced GIF respectiv Progressive JPEG. Dacă o imagine este în acest format atunci la început se va afișa o imagine de calitate inferioară și după ce restul paginii se încarcă, este afișată imaginea întreagă. Folosirea acestor formate are un profund efect psihologic, deoarece navigatorul poate să vadă o imagine completă care se va îmbunătăți treptat și nu trebuie să aștepte pentru o imagine de calitate superioară care ar apărea deodată pe ecran.

Mărimea unei imagini

Deoarece textul este încărcat mult mai repede decât imaginile, cele mai multe browsere Web vor afișa textul înaintea imaginilor. La început browserul nu cunoaște dimensiunile imaginilor și de aceea nu va lăsa loc liber pentru ele. Din această cauză textul se va mișca pe ecran în timp ce imaginile sunt încărcate. Pentru a evita acest lucru comanda **<IMG>** are doi parametri prin care poate fi precizată *lungimea* și *înălțimea* unei imagini. De exemplu, comanda:

```
<IMG SRC="myimage.jpg" WIDTH=200 HEIGHT=100>
```

va rezerva un dreptunghi de dimensiuni 200x100 pentru imaginea din fișierul **myimage.gif**. Dimensiunile transmise ca parametri nu trebuie neapărat să fie egale cu dimensiunile reale ale imaginii, browser-ul mărind sau micșorând imaginea astfel încât ea să ocupe întreg dreptunghiul precizat. Folosirea acestor parametri poate avea rezultate neplăcute deoarece imaginea ar putea fi deformată prin mărirea sau micșorarea disproporționată a celor două dimensiuni.

Imagini Preview

Un alt parametru al comenzii **<IMG>** este **LOWSRC**. Acesta oferă posibilitatea ca la început să fie afișată o imagine, iar după încărcarea restului paginii această imagine să fie înlocuită cu alta de calitate superioară. Un exemplu ar fi:

```
<IMG SRC="highqlty.jpg" LOWSRC="lowqlty.jpg">.
```

De obicei imaginea "preview" este un JPEG cu un grad de compresie foarte mare sau o imagine care conține un număr foarte mic de culori (16, 4 sau 2).

Animații grafice

Metodele avansate de introducere a animațiilor în paginile Web vor fi tratate în unul din episoadele următoare. Există totuși o metodă simplă de a include animații într-o pagină Web și anume folosirea GIF-urilor animate care pot fi introduse la fel de ușor ca orice altă imagine.

Culori

În mod normal o pagină Web are niște culori predefinite pentru fond, text și link-uri. Aceste culori pot fi schimbate cu ajutorul unor parametri ai tag-ului **BODY**. O comandă:

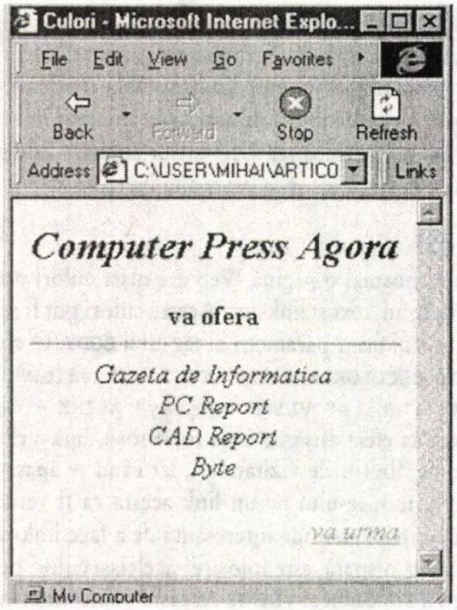
```
<BODY BGCOLOR = "red" TEXT = "yellow"
LINK = "white" VLINK = "gray" ALINK = "green">
```

va avea ca efect afișarea unui fond roșu, link-urile nevizitate albe, link-urile vizitate gri, iar când se apasă butonul stâng al mouse-ului pe un link acesta va fi verde pentru scurt timp. O metodă interesantă de a face link-urile deja vizitate să dispară, este folosirea aceleiași culori pentru parametrii **BGCOLOR** și **VLINK**. Aceasta poate însă nemulțumi unii navigatori care ar dori să revadă o pagină deja vizitată.

Dacă valoarea unui parametru este de tipul `#RRGGBB`, atunci va fi definită o nouă culoare. `RR`, `GG` și `BB` sunt numere hexazecimale de două cifre și reprezintă valorile pentru componenta roșie, verde și albastră a culorii. De exemplu, un roșu deschis va corespunde valorii `#FF0000`, verde închis va fi `#004000`, iar un gri de intensitate medie va fi `#808080`. Iată un exemplu care folosește noțiunile prezentate anterior:

```
<html>
<head>
<title> Culori </title>
</head>

<body bgcolor = "#FFFF80" text = "#400040"
link = "#FF8000" vlink ="#FFFF80">
<div align = "center">
<h2> <i>
Computer Press Agora
</i> </h2>
<b> va ofera </b>
<br> <hr>
<i> Gazeta de Informatica <br>
PC Report <br>
CAD Report <br>
Byte <br>
</div>
<p align = "right">
<a href = "next.htm"> va urma </a>
</body>
</html>
```

La încărcarea acestui fișier va apărea următoarea pagină Web:

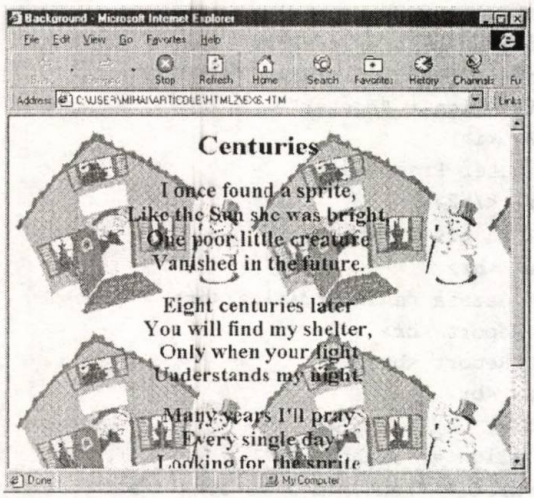


Imagini Background

Majoritatea paginilor Web au ca fond o anumită imagine. Specificarea unei imagini ca *background* se poate face cu ajutorul parametrului **BACKGROUND** al *tag*-ului **<BODY>**. Sintaxa este:

```
<BODY BACKGROUND = "background.jpg">.
```

Dacă imaginea nu va acoperi întreaga fereastră în care este afișată pagina Web, părțile libere vor fi acoperite cu copii ale imaginii. O pagină Web cu background este următoarea:



Fișierul HTML corespunzător este:

```
<html>
<head>
<title> Background
</title>
</head>
<body background = "back.gif">
<div align = "center">
<b> <basefont size = 5>
<h1> Centuries </h1>
I once found a sprite,<br>
Like the Sun she was bright,<br>
One poor little creature<br>
Vanished in the future.<p>
Eight centuries later<br>
You will find my shelter,<br>
Only when your light<br>
Understands my night.<p>
Many years I'll pray<br>
Every single day<br>
Looking for the sprite<br>
I have lost that night.<p>
Nine centuries later<br>
Delivering a letter<br>
Another sprite I found.<br>
</b>
</div>
</BODY>
</HTML>
```

Dacă veți folosi imagini background, trebuie să aveți grijă ca textul care va fi afișat peste background să fie vizibil. Încercați să folosiți pentru text o culoare care nu apare deloc în background sau care apare de foarte puține ori.

În loc de concluzie

După parcurgerea acestor două episoade ale serialului puteți proiecta deja o pagină Web care să-și merite numele. În episoadele următoare vor fi prezentate câteva modalități care vor îmbunătăți calitatea paginilor dumneavoastră. Veți afla noutăți despre introducere de secvențe video, tabele, *frame*-uri, sunete, *script*-uri etc. Sperăm că vă vor interesa *applet*-urile care pot da o notă de profesionalism unei pagini Web.

Mihai Scorțaru este student în anul I al Facultății de Automatică și Calculatoare, Universitatea Tehnică Cluj. Poate fi contactat prin e-mail la adresa: scort@licj.soroscj.ro

OBERON: Structuri generice

Mircea Sârbu

După ce în numărul trecut am trecut în revistă câteva dintre noțiunile fundamentale ale programării obiectuale și modul în care sunt acestea implementate în Oberon, a venit vremea să ne convingem de posibilitățile pe care această tehnică le poate deschide. Ca exemplificare am ales, desigur, structurile înlănțuite.

Structurile dinamice de date sunt o permanentă provocare pentru un programator. Listele și arborii reprezintă adesea cele mai potrivite soluții pentru rezolvarea unor clase diverse de probleme. Din păcate însă, cele mai multe limbaje de programare furnizează doar instrumente fundamentale pentru manevrarea structurilor dinamice și în cele mai multe cazuri se pot înlănțui doar structuri omogene.

Exemplul pe care vi-l propun constă în construirea unui modul care să furnizeze instrumentarul necesar pentru manevrarea unor structuri dinamice complexe și ne-omogene. Ideea este ca acest modul să implementeze o structură de listă înlănțuită generică, a cărei noduri pot conține orice fel de informații. Mai mult, o aceeași listă poate să cuprindă noduri diverse. Odată acest modul realizat, orice modul client poate să folosească structurile înlănțuite fără să le mai implementeze funcționalitatea, indiferent de natura informației care trebuie reprezentată

Înainte de a începe, vă reamintesc că vom folosi o implementare a limbajului Oberon-2 numită *Component Pascal*. Textul sursă va fi cuprins în textul articolului. Considerați explicațiile din corpul de text ca fiind comentarii.

Modul de reprezentare

Ideea reprezentării pe care v-o propun provine din limbajul *Lisp*, un limbaj pentru care întregul univers este format din *atomi* și *liste*. Un atom poate să fie o informație oarecare sau poate să fie o listă. O listă se definește recursiv astfel: O listă poate fi vidă sau poate fi formată dintr-un atom urmat de o listă. În felul acesta, un nod al listei este format întotdeauna dintr-o pereche de pointeri, numiți prin tradiție *car* și *cdr*. Pointerul *car* punctează un atom, reprezentând informația purtată de nod. Pointerul *cdr* este rădăcina listei celorlalte noduri. Dacă nodul este ultimul din listă, atunci *cdr* ia o valoare specială, numită *NIL*.

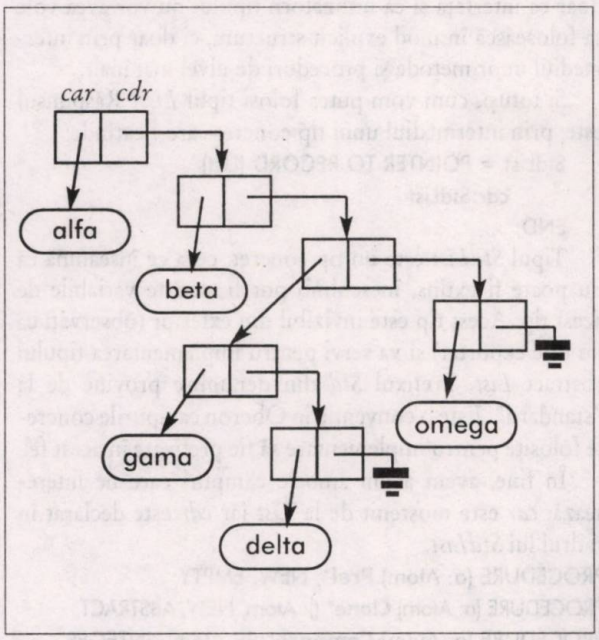
Să considerăm, pentru exemplificare, că atomii reprezintă numere întregi, șiruri de caractere și caractere singulare. Ca o notație "externă", utilă pentru exemplificări, vă propun să reprezentăm listele între paranteze drepte (ca în *Logo*). Astfel, o listă cuprinzând cuvintele "alfa", "beta" și "gama" va arăta cam așa:

```
[ alfa beta gama ]
```

Deoarece am menționat că atomii pot fi și ei liste, iată un exemplu mai puțin simplist:

```
[ alfa beta [ gama delta ] omega ]
```

În figura de mai jos puteți urmări reprezentarea acestei liste prin perechile de pointeri *car* (în stânga) și *cdr* (în dreapta). Surprinza este că această reprezentare nu este liniară, ci arborescentă.



Să încercăm să realizam ceva asemănător în Oberon.

```
MODULE MirLists;
    IMPORT Out; (* Doar pentru testare *)
    CONST equal* = 0; less* = -1; greater* = 1; different* = 9;
```

Am denumit modulul *MirLists* deoarece este practica recomandabilă ca fiecare realizator de programe (firmă sau persoană) să folosească un anumit prefix propriu.

Modulul *Out* permite scrierea în fereastra *Log* și este folosit doar pentru testare.

Deoarece Oberon nu dispune de tipuri enumerate, în mod curentă se folosesc constante întregi prin intermediul unor mnemonice. Aceste constante sunt exportate, deci vor apărea în interfața modulului și, desigur, în documentație. Ele sunt folosite pentru a obține rezultatul unor comparații.

```
TYPE
   Atom* = POINTER TO ABSTRACT RECORD END;
   List* = POINTER TO ABSTRACT RECORD (Atom)
         car: Atom
   END;
```

Iată și declarația tipurilor fundamentale în reprezentarea pe care am propus-o. Tipul *Atom* nu poartă nici o informație și este declarat *ABSTRACT*. Aceasta înseamnă că în acest modul nu pot fi alocate variabile de acest tip. În schimb, poate fi extins. De fapt, tipul *Atom* nu este altceva decât o bază pentru extensie. Orice structuri de informații vom dori să stocăm în listă, vor trebui construite prin extinderea tipului *Atom*.

Tipul *List* este declarat ca extensie a tipului *Atom* tocmai pentru a putea reprezenta structuri de genul celei din figură. De fapt, lista este un atom ceva mai special. Ceea ce poate să pară surprinzător este faptul că și tipul *List* este abstract (deci nu pot fi alocate variabile de acest tip). Un alt aspect interesant este faptul că nu este exportat câmpul *car* iar câmpul *cdr* pur și simplu nu apare în articol.

Explicația constă în faptul că *List* este un tip folosit doar ca interfață și că utilizatorii tipului nu vor avea voie să folosească în mod explicit structura, ci doar prin intermediul unor metode și proceduri de nivel mai înalt.

Și totuși, cum vom putea folosi tipul *List*? Răspunsul este: prin intermediul unui tip concret care-l extinde:

```
   StdList = POINTER TO RECORD (List)
         cdr:StdList
   END;
```

Tipul *StdList* este un tip concret, ceea ce înseamnă că nu poate fi extins, în schimb pot fi alocate variabile de acest tip. Acest tip este invizibil din exterior (observați că nu este exportat) și va servi pentru implementarea tipului abstract *List*. Prefixul *Std* din denumire provine de la "standard". Este o convenție în Oberon ca tipurile concrete folosite pentru implementare să fie prefixate în acest fel.

În fine, avem acum ambele câmpuri care ne interesează: *car* este moștenit de la *List* iar *cdr* este declarat în cadrul lui *StdList*.

```
PROCEDURE (a: Atom) Prel*, NEW, EMPTY;
PROCEDURE (a: Atom) Clone* (): Atom, NEW, ABSTRACT;
PROCEDURE (a: Atom) Compare* (b: Atom): INTEGER,
                              NEW, ABSTRACT;
```

Iată și câteva metode pentru *Atom*. Spre deosebire de cele pentru *List*, acestea nu vor fi implementate în acest modul. Este de altfel imposibil, deoarece ele trebuie să ofere o funcționalitate specifică extensiilor tipului *Atom*. În schimb, prin specificarea acestor metode abstracte îl obligăm pe utilizatorul modulului nostru să le implementeze pentru fiecare extensie a tipului *Atom*. Ele sunt folosite explicit în acest modul, deoarece în orice modul client ele vor fi implementate cu siguranță. Observați că utilizarea acestor metode înseamnă, practic, apelarea unui cod care încă nu există!

Procedura *Prel* este utilizată pentru a specifica o procesare ce trebuie aplicată tuturor nodurilor unei liste. Vom vedea însă că utilizatorul va putea să definească astfel de prelucrări și în modulul client. Metoda nu este abstractă, ci *EMPTY* (vidă). Diferența este că procedura poate fi apelată fără a produce o eroare (e adevărat că nici nu face nimic...).

Metoda *Clone* este concepută pentru a produce o copie (o clonă) a obiectului. În general toate tipurile generice dispun de o procedură de acest gen.

Metoda *Compare* va returna o valoare întreagă reprezentând rezultatul comparației dintre atomul *a* și atomul *b*. Pentru a nu reține diverse numere, vom folosi constantele mnemonice *equal, greater, less* și *different*.

Urmează acum metodele și procedurile tipului *List*. Le-am împărțit în mai multe categorii. Vom începe cu "constructorii".

```
PROCEDURE (l: List) Clone* (): List, ABSTRACT;
PROCEDURE (l: List) FPut* (a: Atom): List, NEW, ABSTRACT;
PROCEDURE (l: List) LPut* (a: Atom): List, NEW, ABSTRACT;
PROCEDURE New* (car, cdr: Atom): List;
   VAR l: StdList;
BEGIN
   ASSERT( (car=NIL) & (cdr = NIL) OR (car # NIL), 21);
   NEW(l); l.car := car;
   IF cdr = NIL THEN l.cdr := NIL
   ELSIF cdr IS List THEN l.cdr := cdr(StdList);
   ELSE NEW(l.cdr); l.cdr.car := cdr; l.cdr.cdr := NIL
   END;
   RETURN l
END New;
```

După cum se vede (din faptul că nu apare marcajul *NEW*), metoda *Clone* este o redefinire a metodei corespunzătoare a tipului de bază, *Atom*. După cum bănuiți, ea va fi apelată pentru a produce o clonă a listei. Pentru a obține această clonă, în implementare se va recurge la clonarea atomilor componenți.

Metoda *FPut* va adăuga în fața listei un nod, cuprinzând atomul furnizat ca parametru. O precondiție pe care trebuie să o asigure clienți modulului este ca atomul să nu fie *NIL* (altfel o aserțiune va opri programul), așa cum vom vedea la implementare.

Metoda *LPut* este similară, cu diferența că nodul este adăugat la sfârșitul listei.

Procedura *New* nu este o metodă, ci o funcție independentă. Am plasat-o aici pentru că joacă un rolul care în alte limbaje obiectuale este numit "constructor" (după cum am menționat în articolul din numărul trecut, în Oberon nu există constructori propriu-ziși, deoarece înainte de alocare nu poate fi apelată nici o metodă). Această funcție este extrem de importantă și este cheia întregii construcții, așa că mă voi opri mai mult asupra ei.

Primul lucru pe care trebuie să-l observăm este că ea nu returnează de fapt o valoare de tipul *List*, ci una de tipul

StdList, care este o extensie concretă a tipului *List*. Utilizatorul nu va ști însă acest lucru, deoarece oriunde poate fi folosit tipul de bază, se poate folosit orice extensie a acestuia. Altfel, nu vor exista niciodată variabile de tipul *List*. Când, în modulul client, va fi invocată o metodă a tipului *List*, executivul Oberon va examina variabila pe care utilizatorul a declarat-o de tip *List* și va constata că tipul dinamic este *StdList*. În consecință, prin mecanismul de legare dinamică, va executa metoda corespunzătoare a acestui tip. Așa se explică "exportul de interfață" specific limbajului Oberon.

Funcția *New* poate fi folosită pentru a crea liste de mai multe forme. Dacă amândoi parametri sunt *NIL*, se va crea o listă vidă. De remarcat că lista vidă o vom reprezentata printr-un nod în care ambii pointeri (*car* și *cdr*) au valoarea *NIL*. Dacă primul parametru (*car*) este diferit de *NIL* iar al doilea este *NIL*, atunci lista creată va avea un singur nod, corespunzător valorii furnizate de acest parametru. Dacă și cel de-al doilea parametru (*cdr*) este diferit de *NIL*, atunci apar două posibilități. Dacă tipul dinamic al atomului *cdr* este *List* (sau o extensie a acestuia, în speță *StdList*) atunci această listă va fi adăugată după primul element (observați că se folosește un *type guard*, deoarece tipul static al parametrului *cdr* este *Atom*). Altfel, lista formată va avea două noduri, corespunzătoare celor doi parametri. În fine, dacă primul parametru este *NIL* iar al doilea este diferit de *NIL* atunci se produce o eroare (este situația pe care o depistează aserțiunea).

Urmează acum un set de metode pe care le voi numi "selectori":

```
PROCEDURE (l: List) First* (): Atom, NEW, ABSTRACT;
```

Returnează primul element al listei.

```
PROCEDURE (l: List) ButFirst* (): List, NEW, ABSTRACT;
```

Returnează lista fără primul element.

```
PROCEDURE (l: List) Last* (): Atom, NEW, ABSTRACT;
```

Returnează ultimul element al listei.

```
PROCEDURE (l: List) ButLast* (): List, NEW, ABSTRACT;
```

Returnează lista fără ultimul element.

```
PROCEDURE (l: List) Remove* (a: Atom): List, NEW,
                                                ABSTRACT;
```

Returnează lista, din care au fost îndepărtate elementele corespunzătoare atomului furnizat ca parametru.

```
PROCEDURE (l: List) Item* (index: INTEGER): Atom, NEW,
                                                ABSTRACT;
```

Returnează elementul aflat pe o anumită poziție în listă. Indexul să fie pozitiv și mai mic decât lungimea listei.

O precondiție comună tuturor selectorilor: lista nu trebuie să fie vidă. Dacă lista conține un singur nod, atunci *ButFirst* și *ButLast* vor returna lista vidă.

O grupă slab reprezentată este cea a "modificatorilor", formată dintr-o singură metodă:

```
PROCEDURE (l: List) SetFirst* (a: Atom), NEW, ABSTRACT;
```

Este vorba despre schimbarea primului valorii din primul nod al listei cu cea furnizată ca parametru. Evident, lista nu trebuie să fie vidă iar atomul nu trebuie să fie *NIL* (în această reprezentare nu are sens un nod care să nu poarte o informație). Este de remarcat că aceasta este singura metodă care afectează la modul direct lista, toate celelalte acționând asupra unor clone.

Urmează "predicatele". Primul dintre ele este implementat chiar în tipul *List*, exemplificând astfel că un tip abstract poate să dispună de metode concrete.

```
PROCEDURE (l: List) IsEmpty* (): BOOLEAN, NEW;
BEGIN
   IF (l.car = NIL) & ( l(StdList).cdr = NIL) THEN
      RETURN TRUE
   ELSE RETURN FALSE END
END IsEmpty;
```

Metoda returnează *TRUE* dacă lista este vidă. Observați că tipul static *List* nu dispune de câmpul *cdr*. Pentru a putea fi referit, variabila trebuie prevăzută cu un gardian de tip care să o facă de tip *StdList*. Operația poate fi făcută fără precauții deoarece știm că orice listă are tipul dinamic *StdList*.

```
PROCEDURE (l: List) IsMember* (a: Atom): BOOLEAN, NEW,
                                                ABSTRACT;
```

Returnează *TRUE* dacă atomul furnizat ca parametru corespunde (cel puțin) unui nod al listei. Metoda se bazează pe comparația atomilor, implementată în modulele client.

```
PROCEDURE (l: List) IsEqual* (list: List): BOOLEAN, NEW,
                                                ABSTRACT;
```

Returnează *TRUE* dacă listele sunt egale. Pentru aceasta, ele trebuie să aibă același număr de elemente și atomii corespunzători acestora să fie (în ordine) egali. Se bazează de asemenea pe comparația atomilor.

```
PROCEDURE (l: List) IsSame* (list: List): BOOLEAN, NEW,
                                                ABSTRACT;
```

Spre deosebire de *IsEqual*, metoda *IsSame* returnează *TRUE* dacă *l* și *list* indică aceeași listă (în sensul că o modificare într-una "se vede" și în cealaltă). Este clar că *IsSame* implică *IsEqual*.

Urmează o ultimă clasă de metode, pe care le putem numi "interogări":

```
PROCEDURE (l: List) Count* (): INTEGER, NEW, ABSTRACT;
```

Este o funcție care furnizează numărul de elemente din listă. Pentru lista vidă returnează 0.

```
PROCEDURE (l: List) Member* (a: Atom): List, NEW,
                                                ABSTRACT;
```

O metodă puțin mai ciudată: dacă există un nod corespunzător valorii atomului furnizat la parametru, atunci întoarce o copie a bucății de listă care începe de la primul nod găsit până la sfârșitul listei. Dacă nodul nu există, returnează lista vidă.

În fine, urmează câteva metode "utilitare":

```
PROCEDURE (l: List) Traverse*, NEW, ABSTRACT;
```

Metoda este utilă pentru aplicarea unei anumite procesări tuturor elementelor unei liste, inclusiv listelor interioare.

```
PROCEDURE (l: List) Prel*, EXTENSIBLE;
BEGIN l.Traverse END Prel;
```

Procedură implementată (și marcată ca extensibilă)

care permite parcurgerea listei. Implementarea acestei proceduri este obligatorie, deoarece în caz contrar tipul *List* moștenește procedura abstractă corespunzătoare de la tipul *Atom*.

```
PROCEDURE (l: List) Compare* (a: Atom): INTEGER;
BEGIN
   IF (a IS List) & (l.IsEqual(a(StdList))) THEN RETURN equal
   ELSE RETURN different END
END Compare;
```

Situația este identică cu cea a procedurii *Prel*. De remarcat că este interesant doar cazul în care se furnizează ca argument o listă și că nu există o relație de ordine între elemente eterogene (pot să fie doar egale sau diferite).

Implementarea

Restul sursei o reprezintă implementarea tipului *StdList*. Comentariile vor fi mai puțin necesare, deoarece procedurile sunt scurte și destul de clare pentru cititorii familiarizați cu limbajul Pascal.

Implementarea nu a vizat atât performanța, cât mai cu seamă claritatea. Am folosit atât procedee iterative cât și procedee recursive. Ceea ce este important este că această implementare poate fi modificată oricând (în sensul sporirii performanței), fără ca schimbarea să afecteze clienții modulului.

Procedurile sunt redefiniri ale metodelor abstracte ale tipului *List* (exceptând cele marcate cu *NEW*, care sunt doar proceduri ajutătoare). Se remarcă în cazul funcțiilor covarianța: metodele care pentru *List* returnau un pointer la *List*, aici returnează pointeri la *StdList*. Este posibilă și implementarea fără utilizarea covarianței (funcțiile să întoarcă pointeri de tip *List*).

```
PROCEDURE (l: StdList) Clone (): StdList;
   VAR p: StdList;
BEGIN
   NEW(p);
   IF ~l.IsEmpty() THEN
         p.car := l.car.Clone();
         IF l.cdr # NIL THEN p.cdr := l.cdr.Clone() END;
   END;
   RETURN p
END Clone;

PROCEDURE (l: StdList) FindLast (): StdList, NEW;
   VAR p: StdList;
BEGIN
   p := l;
   IF ~l.IsEmpty() THEN
         WHILE p.cdr # NIL DO p := p.cdr  END;
   END;
   RETURN p
END FindLast;
PROCEDURE (l: StdList) FPut (a: Atom): StdList;
   VAR p: List;
BEGIN
   ASSERT (a # NIL, 21);
   IF l.IsEmpty() THEN p := New(a, NIL)
   ELSE p := New(a, l) END;
   RETURN p(StdList)
END FPut;

PROCEDURE (l: StdList) LPut (a: Atom): StdList;
   VAR p: List; r: StdList;
BEGIN
   ASSERT (a # NIL, 21);
   IF l.IsEmpty() THEN p := New(a, NIL)
   ELSE
         p := l.Clone(); r := p(StdList).FindLast();
         NEW(r.cdr); r.cdr.cdr := NIL; r.cdr.car := a;
   END;
   RETURN p(StdList)
END LPut;

PROCEDURE (l: StdList) First (): Atom;
BEGIN
   ASSERT (~l.IsEmpty(), 21);
   RETURN l.car.Clone()
END First;

PROCEDURE (l: StdList) ButFirst (): StdList;
BEGIN
   ASSERT (~l.IsEmpty(), 21);
   IF l.cdr # NIL THEN RETURN l.cdr.Clone()
   ELSE RETURN New(NIL, NIL)(StdList) END
END ButFirst;

PROCEDURE (l: StdList) Last (): Atom;
BEGIN
   ASSERT (~l.IsEmpty(), 21);
   RETURN l.FindLast().car.Clone()
END Last;

PROCEDURE (l: StdList) ButLast (): StdList;
   VAR p, r: StdList;
BEGIN
   ASSERT (~l.IsEmpty(), 21);
   IF l.cdr = NIL THEN RETURN New(NIL, NIL)(StdList) END;
   p := l.Clone()(StdList); r := p;
   WHILE r.cdr.cdr # NIL DO r := r.cdr END;
   r.cdr := NIL;
   RETURN p
END ButLast;

PROCEDURE (l: StdList) Remove (a: Atom): StdList;
   VAR p: StdList;
BEGIN
   IF l.IsEmpty() THEN RETURN New(NIL, NIL)(StdList) END;
   NEW(p);
   IF l.car.Compare(a) # equal THEN
         p.car := l.car.Clone() END;
   IF l.cdr # NIL THEN
         p.cdr := l.cdr.Remove(a) (StdList) END;
   RETURN p
END Remove;

PROCEDURE (l: StdList) Item (index: INTEGER): Atom;
   VAR k, n: INTEGER; p: StdList;
```

```
BEGIN
    ASSERT(index > 0, 21);
    n := l.Count(); ASSERT(index <= n, 22);
    k := 1; p := l;
    WHILE k < index DO p := p.cdr; INC(k) END;
    RETURN p.car.Clone()
END Item;

PROCEDURE (l: StdList) SetFirst (a: Atom);
BEGIN
    ASSERT(~l.IsEmpty(), 21); ASSERT(a # NIL, 22);
    l.car := a
END SetFirst;

PROCEDURE (l: StdList) IsMember (a: Atom): BOOLEAN;
    VAR p: StdList;
BEGIN
    ASSERT(a # NIL, 21);
    IF l.IsEmpty() THEN RETURN FALSE END;
    IF l.car.Compare(a) = equal THEN RETURN TRUE
    ELSIF l.cdr = NIL THEN RETURN FALSE
    ELSE RETURN FALSE OR l.cdr.IsMember(a) END
END IsMember;

PROCEDURE (l: StdList) IsSame (list: List): BOOLEAN;
BEGIN RETURN l = list END IsSame;

PROCEDURE (l: StdList) IsEqual (list: List): BOOLEAN;
BEGIN
    IF l = list THEN RETURN TRUE
    ELSIF l.IsEmpty() # list.IsEmpty() THEN RETURN FALSE
    ELSIF (l.car # NIL) & (list.car # NIL) &
                (l.car.Compare(list.car) # equal) THEN
        RETURN FALSE
    ELSE RETURN TRUE & l.cdr.IsEqual(list(StdList).cdr) END
END IsEqual;

PROCEDURE (l: StdList) Count (): INTEGER;
    VAR i: INTEGER; p: StdList;
BEGIN
    IF l.IsEmpty() THEN RETURN 0
    ELSE
        p := l; i := 1;
        WHILE p.cdr # NIL DO INC(i); p := p.cdr  END;
        RETURN i
    END
END Count;

PROCEDURE (l: StdList) Member (a: Atom): StdList;
BEGIN
    ASSERT (a # NIL, 21);
    IF ~l.IsMember(a) THEN RETURN New(NIL, NIL)(StdList)
    ELSIF l.car.Compare(a) = equal THEN
        RETURN l.Clone()
    ELSE RETURN l.cdr.Member(a) END
END Member;
```

Metoda *Traverse* este utilizată aici în conjuncție cu metoda *Prel* a diverselor extensii posibile ale tipului *Atom* pentru a vizualiza conținutul listei. Remarcați recursivitatea indirectă: dacă un element al listei este o alta lista, atunci apelul *p.car.Prel* va avea ca efect legarea dinamică a metodei *Prel* a tipului *List*, care la rândul ei va invoca *Traverse*.

```
PROCEDURE (l: StdList)Traverse;
    VAR p: StdList;
BEGIN
    Out.String("[ ");
    IF ~l.IsEmpty() THEN
        p := l;
        REPEAT  p.car.Prel; p := p.cdr
        UNTIL p = NIL
    END;
    Out.String("] ")
END Traverse;

END MirLists.
```

Cum se utilizează

Pentru utilizarea modulului *MirLists* voi furniza un exemplu minimal, dar suficient (sper) pentru a sugera diversele aplicații posibile.

```
MODULE MirTLists;

    IMPORT L := MirLists, Out;

    TYPE
        ChElem = POINTER TO RECORD (L.Atom)
                ch: CHAR
        END;

        StrElem = POINTER TO RECORD (L.Atom)
                str: ARRAY 256 OF CHAR
        END;
```

Se folosesc două tipuri de elemente: unul va cuprinde doar un caracter iar celălalt un șir de caractere. Pentru a putea fi înlănțuite, trebuie să fie extensii ale tipului *Atom* din modulul *MirLists* (numit, pe scurt, *L*).

Fiind extensii ale tipului *Atom*, ambele tipuri moștenesc metodele acestuia, pe care trebuie să le implementeze (deoarece erau declarate abstracte în modulul *L*).

```
PROCEDURE (a:ChElem) Clone(): ChElem;
    VAR c: ChElem;
BEGIN NEW(c); c.ch := a.ch; RETURN c END Clone;

PROCEDURE (a: ChElem) Compare (b: L.Atom): INTEGER;
BEGIN
    IF (b IS ChElem) & (a.ch = b(ChElem).ch) THEN
        RETURN L.equal
    ELSE RETURN L.different END
END Compare;

PROCEDURE (a: ChElem) Prel;
BEGIN Out.Char(a.ch); Out.Char(' ') END Prel;

PROCEDURE (a: StrElem) Prel;
BEGIN Out.String(a.str); Out.Char(' ') END Prel;

PROCEDURE (a: StrElem) Clone(): StrElem;
    VAR c: StrElem;
BEGIN NEW(c); c.str := a.str$; RETURN c END Clone;

PROCEDURE (a: StrElem) Compare (b: L.Atom): INTEGER;
BEGIN
    IF (b IS StrElem) & (a.str = b(StrElem).str$) THEN
        RETURN L.equal
    ELSE RETURN L.different END
END Compare;
```

În metodele *Compare* s-ar fi putut folosi ordinea implicită (lexicografică pentru șiruri de caractere), ceea ce ar fi permis și folosirea unor liste ordonate. Procedura *Show* este folosită doar pentru a scrie (cu concursul metodelor *Prel*) în fereastra *Log* listele într-o formă lizibilă.

```
PROCEDURE Show(l: L.List; name: ARRAY OF CHAR);
BEGIN
  Out.Ln; Out.String(name + " = "); l.Traverse;
END Show;

PROCEDURE Do*;
  VAR a, b, c, d : ChElem; e, f: StrElem;
  l1, l2, l3: L.List;
BEGIN
  NEW(a); a.ch := "a"; NEW(b); b.ch := "b";
  NEW(c); c.ch := "c"; NEW(d); d.ch := "d";
  NEW(e); e.str := "alfa"; NEW(f); f.str := "beta";
  l1 := L.New(a, b); l2 := L.New(c, d);
  l3 := L.New(e, NIL);
  l1 := l1.LPut(f); l2 := l1.FPut(l2);
  l3 := l3.LPut(l2.ButLast());
  Show(l1, "l1"); Show(l2, "l2"); Show(l3, "l3");
  l1.SetFirst(e); Show(l1.FPut(l2.First()), "l1");
  IF l1.IsMember(b) THEN
       Out.Ln; Out.String("b este in l1")
  ELSE Out.Ln; Out.String("b NU este in l1") END;
  Show(l1.Member(e), "l1.Member(e)");
END Do;

END MirTLists.
```

Rezultatul executării procedurii *Do* va afișa în fereastra Log următoarele:

```
l1 = [ a b beta ]
l2 = [ [ c d ] a b beta ]
l3 = [ alfa [ [ c d ] a b ] ]
l1 = [ [ c d ] alfa b beta ]
b este in l1
l1.Member(e) = [ alfa b beta ]
```

Pare corect.

Moștenire și compunere

În articolul de luna trecută am amintit, printre altele, că Oberon nu încurajează moștenirea implementării și că este preferabilă compunerea. Poate ar merita să încheiem acest articol cu o scurtă exemplificare relativă la acest subiect.

Să presupunem că am vrea să implementăm un mecanism generic de stivă. ***Stiva*** este un caz particular de listă, în care inserarea și ștergerea elementelor se fac doar într-unul dintre capetele listei (numit vârful stivei) prin operațiile numite de regulă *Push* și respectiv *Pop*.

Fiind un caz special de listă, ne-am putea gândi la extinderea tipului *List*. Este posibil, însă apare o mică problemă: tipul pe care vrem să-l definim (să-l numim *Stack*) va moșteni doar interfața tipului *List*, nu și implementarea (deoarece aceasta este realizată printr-o altă extensie, *StdList*). Asta înseamnă că va trebui să implementăm din nou toate metodele, deoarece în definiția tipului *List* au fost declarate abstracte.

Avem în schimb varianta mai convenabilă a compoziției. Și în acest caz, un exemplu este mai expresiv decât o descriere mai mult sau mai puțin formală:

```
MODULE MirStacks;
  IMPORT L := MirLists;

  TYPE
       Stack* = POINTER TO ABSTRACT RECORD
                  l: L.List;
                  count-: INTEGER;
       END;
```

Tipul *Stack* cuprinde o listă și un contor (*count*) al elementelor din stivă; Acesta poate fi consultat înainte de operațiile de extragere (*Pop*). Vom avea și un tip concret pentru implementare:

```
  StdStack = POINTER TO RECORD (Stack) END;

PROCEDURE (s: Stack) Push* (a: L.Atom), NEW,ABSTRACT;
PROCEDURE (s: Stack) Pop* (): L.Atom, NEW, ABSTRACT;
```

Observați că metoda *IsEmpty* nu face apel la metoda corespunzătoare a tipului *List*, ci se bazează exclusiv pe contor. În rest, metodele tipului *List* pot fi folosite.

```
PROCEDURE (s: Stack) IsEmpty* (): BOOLEAN, NEW;
BEGIN RETURN s.count = 0 END IsEmpty;

PROCEDURE New* (): Stack;
  VAR s: StdStack;
BEGIN
  NEW(s); s.l := L.New(NIL, NIL);
  s.count := 0; RETURN s
END New;

PROCEDURE (s: StdStack) Push (a: L.Atom);
BEGIN s.l := s.l.FPut(a); INC(s.count) END Push;
PROCEDURE (s: StdStack) Pop (): L.Atom;
  VAR p: L.Atom;
BEGIN
  ASSERT ( ~s.l.IsEmpty(), 21);
  DEC (s.count);
  p := s.l.First(); s.l := s.l.ButFirst();
  RETURN p
END Pop;

END MirStacks.
```

Tot compunerea poate fi folosită pentru a suplini moștenirea compusă, fără însă a "beneficia" de complicațiile pe care aceasta le aduce.

Pe curând

Sper că v-am convins că Oberon este un limbaj puternic și expresiv, capabil de performanțe adesea spectaculoase. Dacă, totuși, unele restricții sau forțări vi s-au părut supărătoare, trebuie să țineți seama că Oberon este conceput ca un instrument pentru medii bazate pe componente software, unde siguranța este esențială. Dar, despre aceasta, altădată...

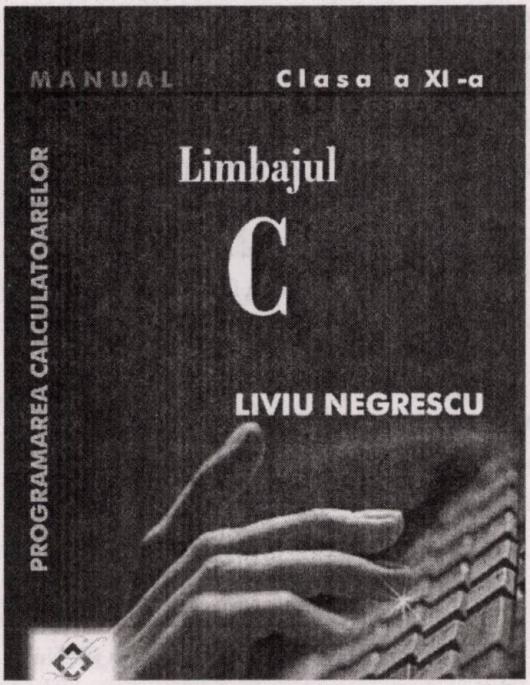
Mircea Sârbu este redactorul șef al revistei PC Magazine Romania. Poate fi contactat prin e-mail la adresa msarbu@agora.ro.

Dacă *încă* nu

EȘTI PREGĂTIT PENTRU MÂINE,

trebuie să mergi la ȘCOALĂ, *trebuie* să ÎNVEȚI, îți *trebuie*

... un MANUAL.



Această carte este concepută în conformitate cu Programa analitică a clasei a XI-a, profilul Informatică. Ea poate fi considerată o introducere elementară în limbajul C. Fiecare noțiune este explicată în mod clar și utilizată în programe simple. De asemenea, capitolele sunt însoțite de exerciții propuse care contribuie la fixarea și înțelegerea noțiunilor întâlnite în capitolele respective.

Cartea se adresează atât elevilor cât și celor interesați să se inițieze în limbajul C care în prezent este unul dintre cele mai utilizate limbaje de programare.

Liviu Negrescu este cercetător principal gr. I la societatea SOFTWARE ITC S.A., filiala Cluj, unde se ocupă cu dezvoltări de programe de sistem și aplicative.
În același timp, desfășoară o activitate didactică prestigioasă de peste 15 ani, la Facultatea de Calculatoare de la Universitatea Tehnică din Cluj.
În decursul anilor a fost titular a mai multe cursuri, ca de exemplu:
- Limbaje de programare și compilatoare;
- Proiectarea translatoarelor etc.

Liviu Negrescu are o serie de lucrări științifice publicate în diferite reviste de specialitate. În colaborare cu alți cercetători de la SOFTWARE ITC S.A. a implementat sisteme cu aplicații în diferite domenii:
- Construcția de compilatoare;
- Interfețe utilizator;
- Prelucrări de date din domenii economice și medicale.

Liviu Negrescu este autorul mai multor cărți apărute în decursul anilor, cu privire la diferite limbaje de programare (CPL1-subset al limbajului PL/1, CDL-limbaj de descriere al compilatoarelor precum și referitoare la limbajele C și C++. Lucrarea beneficiază de experiența vastă a autorului câștigată atât în activitatea de cercetare cât și în cea didactică.

COMPUTER LIBRIS
AGORA

Pentru comenzi sunați la Computer Libris AGORA [Tel: 064-192422]sau la Computer Press AGORA [Tel: 065-166516; 01-3309281].

Vă place GInfo?

Nume: _____ Prenume: _____ Vârsta: _____

Adresa: _____

Localitate: _____ Cod poștal: _____ Județ: _____ Telefon: _____

Ocupația: elev informatică ☐ elev ☐ student ☐ cadru didactic ☐ altă ocupație ☐ (precizați: _____)

Școala (firma, organizația etc.): _____

Dotări

Folosiți un calculator acasă? Da ☐ Nu ☐

Câte calculatoare sunt instalate în școală: (aprox.) _____

Aveți acces la e-mail: Da ☐ Nu ☐

Aveți acces la WWW: Da ☐ Nu ☐

Cum apreciați dotarea cu tehnică de calcul a școlii: _____

Aprecieri generale

Acordați note de la 1 la 10 pe următoarele criterii:

Structura tematică: ___

Calitatea articolelor: ___

Prezentare grafică: ___

Calitate tipografică: ___

Considerați că prețul este:

prea mare ☐ accesibil ☐ prea mic ☐

De cât timp cunoașteți Gazeta de Informatică (ani)? _____

Cum ați procurat acest număr:

chioșc ☐ abonament ☐ cadou ☐ altfel ☐

În acest număr

Apreciați cu note de la 1 la 10 articolele acestui număr:

TI & C în reforma curriculară ___

Digrafuri caracterizate prin K-formule ___

Înfășurătoarea convexă în 2D ___

Macrouri în Word '97 ___

Programarea plăcii de sunet ___

Lucrul cu șiruri în Java ___

Pagini WEB cu HTML 3.2 ___

Oberon: Structuri generice ___

Articolul care v-a plăcut cel mai mult: _____

De ce? _____

Articolul care v-a plăcut cel mai puțin: _____

De ce? _____

Considerați că ponderea problemelor rezolvate este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea problemelor propuse este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că ponderea subiectelor teoretice este:

prea mică ☐ potrivită ☐ prea mare ☐

Considerați că nivelul de accesibilitate al articolelor este

prea cobotât ☐ potrivit ☐ prea ridicat ☐

În viitor

În următoarele numere așteptați:

mai multe articole de programare? Da ☐ Nu ☐

mai multe articole teoretice? Da ☐ Nu ☐

mai multe articole despre Internet? Da ☐ Nu ☐

mai multe prezentări de programe? Da ☐ Nu ☐

prezentări de jocuri? Da ☐ Nu ☐

articole despre hardware? Da ☐ Nu ☐

V-ar plăcea să citiți:

pagini redactate de elevii unor licee? Da ☐ Nu ☐

lucrări premiate, realizate de elevi? Da ☐ Nu ☐

actualități școlare? Da ☐ Nu ☐

prezentarea unor situri Web? Da ☐ Nu ☐

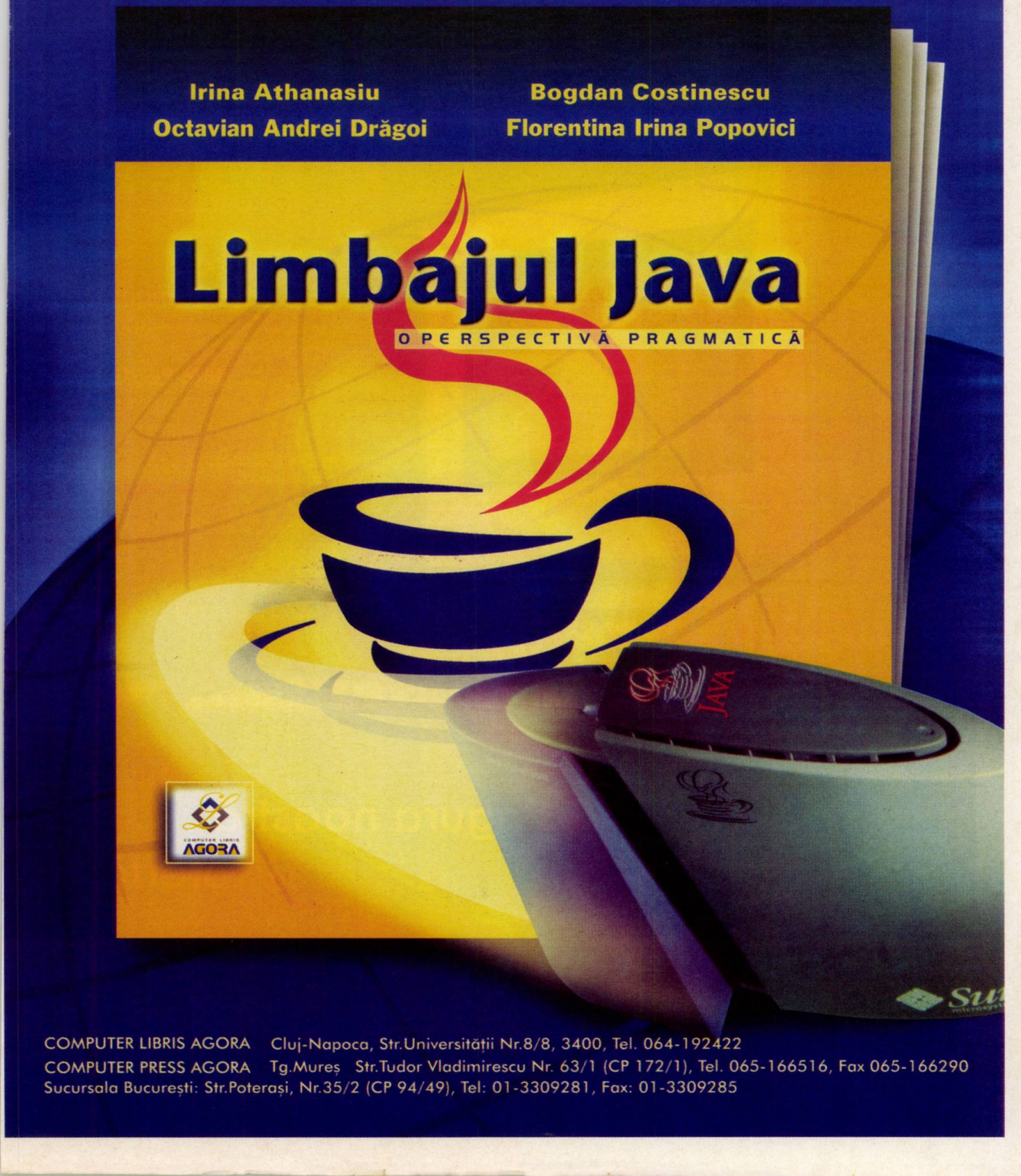
tips & tricks? Da ☐ Nu ☐

Comentarii:

Dacă nu vă este suficient spațiul, folosiți coli albe suplimentare

rtea AGORA • Cartea AGORA • Cartea AGORA • Cartea AGORA • Cartea AGORA • Cartea AGORA • Car

COMPUTER LIBRIS AGORA



COMPUTER LIBRIS AGORA Cluj-Napoca, Str.Universității Nr.8/8, 3400, Tel. 064-192422
COMPUTER PRESS AGORA Tg.Mureș Str.Tudor Vladimirescu Nr. 63/1 (CP 172/1), Tel. 065-166516, Fax 065-166290
Sucursala București: Str.Poterași, Nr.35/2 (CP 94/49), Tel: 01-3309281, Fax: 01-3309285

Ştiţi cât costă un IBM PC?

doar 1100 USD !!!

Dacă pe lângă cei **800** USD cât costă un pachet **LOTUS SmartSuite** mai plătiţi **300** USD, primiţi şi un calculator original **IBM PC300GL**, în următoarea configuraţie:

- procesor INTEL Pentium MMX 200MHz
- 16MB RAM
- 2,1GB HDD
- placă video PCI 1MB VRAM
- monitor IBM G42 14"
- tastatură IBM 104 taste UK
- mouse IBM
- Windows 95 preinstalat, cu licenţă
- licenţă LOTUS SmartSuite

şi nu este singura noastră ofertă !!!

IBM® Business Partner

Microsoft
WHERE DO YOU WANT TO GO TODAY?™
Authorized Distribution Partner/
DSP for Romania

FORTE COMPUTERS
Flexibility.Made in Romania

IBM® Business Partner

Lipscani 102, Bucureşti
Tel: 3122360/6137282/3120948 Fax: 3122630